ACE User Guide (UG070)

Achronix CAD Environment (v9.1)

Copyrights, Trademarks and Disclaimers

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Table of Contents

Preface

About This Guide

This guide is a reference manual for ACE, used for placing, routing, configuring, and debugging Achronix FPGAs. ACE works in conjunction with third-party synthesis and simulation tools to provide a complete design environment for Achronix FPGAs.

This guide consists of the following chapters:

- Getting Started (see page 22) includes an Introduction to ACE and a quick Tutorial.
- Concepts (see page 24) covers all the basic concepts of ACE, and can be considered a reference manual for the various GUI elements.
- Tasks (see page 259) details how to complete various tasks within the GUI, plus provides the related TCL commands.
- TCL Command Reference (see page 509) provides a complete TCL command reference, including syntax.
- Troubleshooting (see page 611) shows a number of common problems and the recommended solutions.
- Revision History (see page 634) lists the changes to each revision of this document.

Related Documents

The latest version of this document (UG070) is available from your Achronix FAE.

The following documents are always available for download at http://www.achronix.com/documentation/

- *ACE Installation and Licensing Guide* (UG002)
- *Bitstream Programming and Debug Interface User Guide* (UG004)
- *Snapshot User Guide* (UG016)
- *Synthesis User Guide* (UG018)

The following supplemental documents, typically available at the Achronix FTP site (login required), should also be consulted for the very latest information:

- *ACE Release Notes* (RN001)

Further documents are available for each fabric family on both the website and FTP site.

Please consult your Achronix FAE for a complete list of documentation relevant to your Achronix products.

Conventions Used in this Guide

Item	Format	Examples
Command-line entries	Formatted with a bold fixed-width font, or in a special code block.	**`$ Open top_level_name.log`** **`Command-line code example`** `$ Open top_level_name.log`
File Names	Formatted with a fixed-width font.	`filename.ext`
GUI buttons, menus, menu or list choices, and radio buttons	Formatted with a variable-width bold font.	Select **File -> Open**, select the desired file, then click **OK** to continue.
Variables	Formatted with italic emphasis and enclosed by the angle brackets < and >.	*<design_dir>*`/output.log`
RTL Names	Formatted with italic emphasis.	*read_clk*
Window and dialog box headings and sub-headings	Heading formatted in quotation marks.	Under "Output Files", select ...
Window and dialog box names	Name uses initial caps.	From the Add Files dialog box, ...

Chapter - 1: Getting Started

Introduction

The Achronix implementation flow uses an industry standard RTL synthesis flow based on Synplify Pro from Synopsis. Working in conjunction with the synthesis tool, Achronix CAD Environment (ACE) provides:

- Placement
- Routing
- Timing Analysis
- Bitstream Generation
- FPGA Configuration
- On-chip Debugging
- Hard/Soft IP Configuration Tools
- Simulation Netlist Generation

ACE Quickstart Tutorial

Start by copying all the files from *<install_dir>* `/Achronix/examples/quickstart/`*<device>* into a new empty directory (*<test_dir>*). Use the *<device>* directory that matches the Target Device implementation option that you select in step 2. Now click the ( ) icon in the upper right corner of the Welcome view to minimize these instructions. Then follow these simple steps to complete your first design in ACE.

1. Create your Project

In the Projects View (see page 125), click the **Create Project** ( ) toolbar button. In the Create Project Dialog (see page 162), enter (or browse to) the path to *<test_dir>* in the Project Directory field. Enter "quickstart" in the Project Name field and click **OK**. You should now see your new project show up in the Projects view.

See Creating Projects (see page 268) or Working with Projects and Implementations (see page 268) for more details.

2. Add your Design Files and Set Implementation Options

In the Projects View, click the "quickstart" project to select it. Now click the **Add Files** ( ) toolbar button. In the Add Source Files Dialog (see page 147), select `quickstart.vma`, `quickstart.pdc`, and `quickstart.sdc` by holding down the **CTRL** key and clicking each item. Now click the **Open** button to add the files to your project. Finally, in the Options view, expand the **Design Preparation** section and select the **Target Device** that matches the set of design files that you copied earlier. You now have a project that is ready to run through the flow!

See Adding Source Files (see page 272) or Working with Projects and Implementations (see page 268) for more details.

3. Run the Flow

In the Flow View (see page 61), click the **Run Flow** ( ) toolbar button. Output from the Flow (see page 221) is shown in the Tcl Console View (see page 144). When the flow is finished running, you see the Flow Steps (see page 221) in the Flow View updated with a green check mark ( ) to indicate success, and all newly generated reports are displayed in the editor area.

See the Flow (see page 221) concept or Running the Flow (see page 280) for more details.

4. Analyze the Results

On the main toolbar, click the **Floorplanner Perspective** ( ) toolbar button. Within this perspective, use the Critical Paths View (see page 48) to analyze critical paths and highlight them in the Floorplanner View (see page 53). Clicking the **Zoom To Path** ( ) toolbar button in the Critical Paths View zooms the Floorplanner View (see page 53) to the path currently selected in the Critical Paths View. Use the Search View (see page 132) and Selection View (see page 136) to locate objects of interest. Clicking the **Zoom To Selection** ( ) toolbar button in the Selection View zooms the Floorplanner View to the objects in the current selection set.

Congratulations!!!

You have successfully completed a design in ACE!

Chapter - 2: Concepts

Workbench

The term Workbench refers to the desktop development environment within ACE. The Workbench aims to achieve seamless tool integration by providing a common platform for the creation, management, and navigation of project resources.

Each Workbench window contains one or more Perspectives (see page 24). Perspectives contain views (see page 31) and editors (see page 26) and control what appears in certain menus and tool bars. More than one Workbench window can exist on the desktop at any given time.

Perspectives

There are many different kinds of information that must be viewed within ACE. Perspectives are used to filter the information into usable, logically consistent groupings. A perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. A perspective defines the initial set and layout of views (see page 31), editors (see page 26), menus, and toolbars in the Workbench (see page 24) window.

For example, the Projects perspective combines views (see page 31) commonly used while managing project source files, while the Floorplanner perspective contains the views that are used while viewing chip layout and floorplanning information. Perspectives are frequently switched while working inside the Workbench (see page 24).

Note

Within the Workbench (see page 24) window, all perspectives share the same set of Editors (see page 26). All editors are usable/visible from all perspectives. Likewise, each of the Views (see page 31) may optionally be used within any perspective, but they are most useful when grouped with the other views from their native perspective. One of the views, the Tcl Console View (see page 144), is a member of all the perspectives.

Projects Perspective

The ( )Projects Perspective allows selecting an active project and implementation, managing the contents and configuration of the active project/implementation, running the Flow (see page 221), and viewing the reports generated by the Flow.

By default, this perspective contains the Projects View (see page 125), Flow View (see page 61), Options View (see page 103), Tcl Console View (see page 144), and the Editor area, which can contain any ACE Editor or Report. The Multiprocess View (see page 83) is also part of this perspective, but is hidden by default.

For more information, see Working with Projects (see page 268), Running the Flow (see page 280), and Using the Tcl Console (see page 301).

Floorplanner Perspective

The ( )Floorplanner Perspective allows viewing and editing the placement and routing of the active project /implementation.

By default, this perspective contains the following:

- Floorplanner View (see page 53)
- Search View (see page 132)
- Selection View (see page 136)
- Critical Paths View (see page 48)
- Critical Path Diagram View (see page 45)
- Netlist Browser View (see page 89)
- Clock Domains View (see page 33)
- Clock Regions View (see page 36)
- Placement Regions View (see page 120)
- Partitions View (see page 117)
- Tcl Console View (see page 144)

For more information on using the views in this perspective, see Viewing the Floorplanner (see page 312), Pre-Placing a Design (see page 318), and Analyzing Critical Paths (see page 324).

> **Caution!**
>
> Unlike all other perspectives, the Floorplanner perspective hides the Editor area. To view editors (see page 26) and reports (see page 227), a different perspective must be selected.

IP Configuration Perspective

The ( )IP Configuration Perspective is used to create and edit IP configuration files (`.acxip`) through the various IP Configuration Editors.

By default, this perspective contains the following:

- Projects View (see page 125)
- IP Libraries View (see page 81)
- IP Diagram View (see page 80)
- IP Problems View (see page 81)
- Outline View (see page 112)
- Tcl Console View (see page 144)
- I/O Designer Toolkit Views (see page 66)

The IP Configuration Perspective also contains the Editor Area, which can contain any ACE Editor or Report.

See Creating an IP Configuration (see page 306) for more details.

2D NoC Performance Perspective

The ( ) 2D NoC Performance Perspective is used to visualize the throughput or congestion of the 2D NoC network by loading a simulation log file produced by the device simulation model (DSM).

By default, this perspective contains the following:

- NoC Performance View (see page 95)
- NoC Time Slice View (see page 100)

Programming and Debug Perspective

The ( ) Programming and Debug Perspective allows interaction with Achronix FPGAs via JTAG through a JTAG pod or embedded JTAG controller device. Downloading the device configuration and debugging will typically happen from here.

By default, this perspective contains the following:

- Snapshot Debugger View (see page 139)
- Download View (see page 51)
- Tcl Console View (see page 144)

The Programming and Debug Perspective also contains the Editor area, which can contain any ACE Editor or Report.

For more information on using this perspective, see Running the Snapshot Debugger (see page 335) and Playing a STAPL File (Programming a Device) (see page 350)

HW Demo Perspective

The ( ) HW Demo Perspective allows observing various aspects of a particular device, by selecting one of the provided demonstration designs from a list. When the demonstration is loaded into the attached board, LED states and DIP switch states (from the board) are displayed and updated in real-time. Internal device state information such as the temperature of the FPGA and power consumption are also displayed.

By default, this perspective contains the following:

- Snapshot Debugger View (see page 139)
- HW Demo View (see page 64)
- Tcl Console View (see page 144)

The HW Demo Perspective also contains the Editor area, which can contain any ACE Editor or Report.

For more information on using this perspective, see Running the HW Demo (see page 361).

Editors

Most Perspectives (see page 24) in the Workbench (see page 24) are comprised of an editor area and one or more views (see page 31). Different editors are associated with different types of files. For example, when a file is opened by double-clicking in the Projects View (see page 125), the associated editor opens in the Workbench. If there is no associated editor for a resource, the Workbench attempts to launch an external editor outside the Workbench. Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor.

Tabs in the editor area indicate the names of resources that are currently open for editing (usually the filename, and the tab's tooltip provides the full path to the file). An asterisk (*) displayed in an editor tab indicates that an editor has unsaved changes. By default, editors are stacked in the editor area, but users may choose to tile (see page 264) them in order to view multiple editors simultaneously. The gray border at the left margin of the editor area may contain icons that flag errors, warnings, or problems detected by the system.

In ACE, the editor area is also used to view the Reports (see page 227) generated by ACE. By default, when ACE is running the Flow (see page 221) in single-process mode, ACE opens HTML versions of the reports in the HTML Report Browser (see page 27) as soon as the report data is generated/updated. When ACE is in Multiprocess mode (via the Multiprocess View (see page 83)), only the Multiprocess Summary Report (see page 238) is automatically opened in the editor area – the other reports must be opened manually through the Projects View (see page 125), or by following the Timing Report hyperlinks for each Implementation (see page 215) found within the Multiprocess Summary Report.

ACE also provides a suite of IP Configuration Editors, organized by fabric family/library, used to instantiate and configure the various IP surrounding the core fabric. See Creating an IP Configuration (see page 306).

> The IP Libraries and IP types displayed within ACE are dynamic and change based on which technology libraries and devices are installed and licensed at each customer site. Therefore, the individual IP Configuration Editors are not documented in this guide, and are instead documented in separate dedicated Achronix documentation. An example would be the *Speedster7t Soft IP User Guide* (UG103).

HTML Report Browser

When HTML versions of generated Reports (see page 227) are opened within ACE, they are displayed within the Editor area using the HTML Report Browser. This is a very limited form of a web browser — it only allows hyperlink traversal, refresh, forward, and back operations. The buttons for Refresh, Back, and Forward are not displayed within the browser itself, but are instead shown in the main (topmost) ACE button-bar.

> **Note**
>
> The HTML Report Browser should not be used to browse the Internet — a dedicated web browser like Firefox would be a much better choice, for both security and performance reasons.

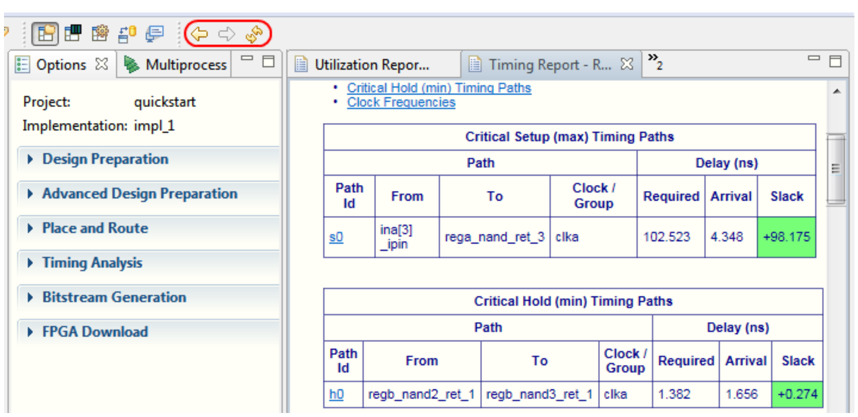


Figure 1: ***HTML Report Browser, Toolbar Buttons Circled in Red***

Table 1: *HTML Report Browser Toolbar Buttons*

Icon	Action	Description
	Back	Returns to the last HTML location viewed.
	Forward	Returns to the HTML location viewed before the **Back** button was selected (the **Forward** button remains disabled until the **Back** button has been pressed).
	Refresh	Refreshes the displayed HTML report to show the current contents of the report file on disk.

Text Editor

Reports, source files, and scripts open in the text editor. The text editor supports typical editing functions, such as insert, delete, copy, cut, and paste.

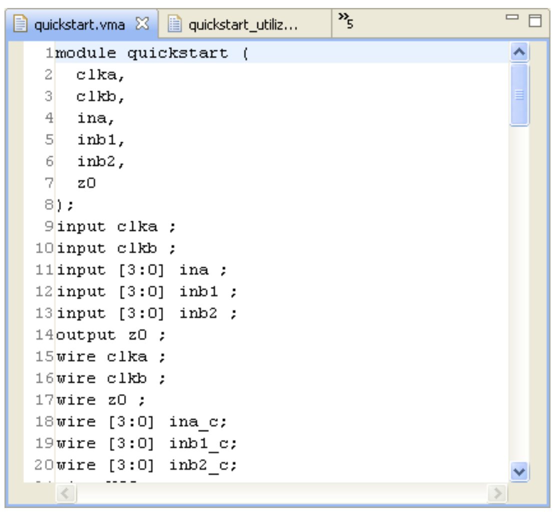


Figure 2: *Text Editor Example*

VCD Waveform Editor

The VCD Waveform Editor does not allow editing a VCD file. It only allows viewing. But since it resides in the same location in the GUI as all the other Editors (see page 26), and it opens whenever a VCD file is selected, it can be thought of as an editor in read-only mode.

The waveform viewer allows examining VCD output in a familiar waveform visualization, displaying how signals change values over time. It is typically used to examine the VCD output that gets generated when Running the Snapshot Debugger (see page 335) (see also: Snapshot Debugger View (see page 139)).

As with familiar waveform editors, the placement of a Marker (here, a pink vertical line) can be manipulated with the mouse in the graphical waveform area, so that the value of all signals at the same instant of time can be seen. Signals can also be re-ordered (an individual signal moved vertically amongst its peers) and individual signals can be hidden or shown. If necessary, signals may be duplicated in the display so that they can be displayed adjacent to multiple peers for ease of value comparisons. It is, of course, possible to change the zoom level of the graphical waveform area if desired.

For each VCD file, the editor remembers signal name ordering, panel sizes, zoom level, and the sample offset between file loads. These are remembered between Snapshot captures within a single session, as well as between ACE sessions.

Note

1. In addition to the graphical waveform view, the raw text content of the VCD file may be viewed. To do so, select the **File Preview** tab at the bottom of the VCD Waveform Editor. To see the graphical waveform representation again, select the **Waveform** tab at the bottom of the VCD Waveform Editor.
2. None of the actions available in the VCD Waveform Editor change the content stored in the VCD file.

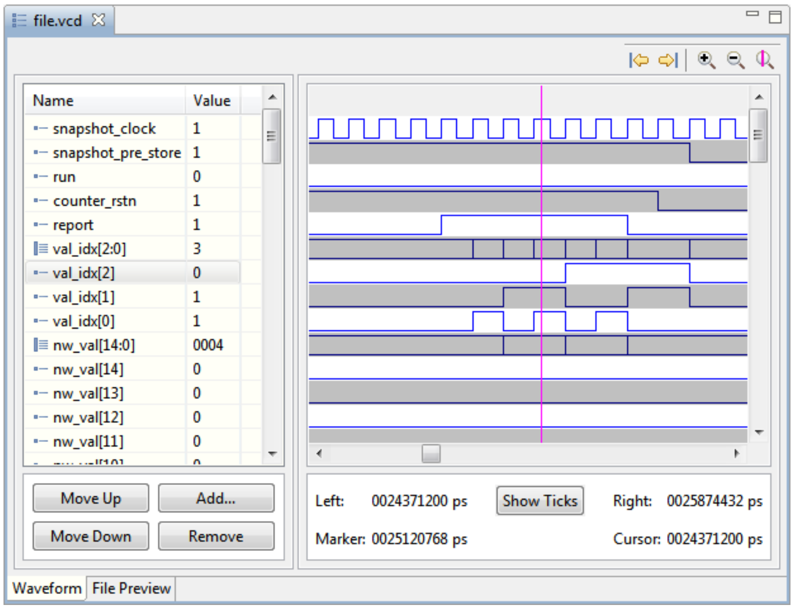


Figure 3: ***VCD Waveform Editor Example***

Table 2: ***VCD Waveform Editor Options***

Option	Description
Signal Value Table	
Name	The name of the signal as stored in the VCD file.
Value	The value of the signal at the Marker's indicated point in time.
Waveform Timing Info	
Left	The time (in ps or tk) indicated by the left edge of the viewable waveform area.
Right	The time (in ps or tk) indicated by the right edge of the viewable waveform area.
Marker	The time (in ps or tk) indicated by the vertical Marker line (in pink, by default) shown in the viewable waveform area.
Cursor	The time (in ps or tk) indicated by the current (or last relevant) mouse cursor position over the viewable waveform area.

Note about Cursor values

When the mouse moves away from the waveform area, the last position is retained by the Cursor value. This ps or tk value does not change until the mouse cursor is again over the waveform, even if the view is scrolled or the zoom factor is changed.

Table 3: ***VCD Waveform Editor Icons***

Icon	Description
	Signal
	Bus

Table 4: ***VCD Waveform Editor Buttons***

Icon	Action	Description
Signal Value Table Buttons		
	Move Up	Moves the currently-selected signals (or buses) one row higher in the table and the waveform area. Signals may also be dragged vertically to new locations with the mouse.
	Move Down	Moves the currently-selected signals (or buses) one row lower in the table and the waveform area. Signals may also be dragged vertically to new locations with the mouse.
	Add...	Opens the Add Signals to Waveform Viewer Dialog (see page 146), which allows previously removed (hidden) signals to be displayed, and allows already-visible signals to be added to the signal list multiple times. (A signal could be duplicated and shown adjacent to multiple associated signals.)
	Remove	Hides the currently selected signal (or bus), temporarily removing it from the table and the waveform area. The hidden signal/bus may be shown again via the **Add...** button.
Waveform Buttons		
	Move Marker to Previous Edge	Moves the vertical pink Marker line in the waveform area to the previous edge for the signal/bus which is currently selected in the signal value table (disabled when no signal is selected in the table).
	Move Marker to Next Edge	Moves the vertical pink Marker line in the waveform area to the next edge for the signal/bus which is currently selected in the signal value table (disabled when no signal is selected in the table).
	Zoom In	Increases the zoom factor in the waveform area, increasing the visible level of detail.

Icon	Action	Description
	Zoom Out	Decreases the zoom factor in the waveform area, decreasing the visible level of detail.
	Zoom to Marker Position	Without changing the zoom factor, scrolls the waveform area horizontally to make the marker visible.
	Show Ticks /Show Times	Toggles the Waveform Timing Info (Left, Right, Marker, and Cursor) between displaying values in Ticks (tk) or Times (ps).

Views

Views support Editors (see page 26) and provide alternative presentations as well as ways to navigate the information in the Workbench (see page 24). For example, the Projects View (see page 125) displays Projects (see page 215), Implementations (see page 215), and their related file-based resources.

All views have their own context menu showing ways to alter the location or presentation of the view. Simply right-click the view tab to display the menu.

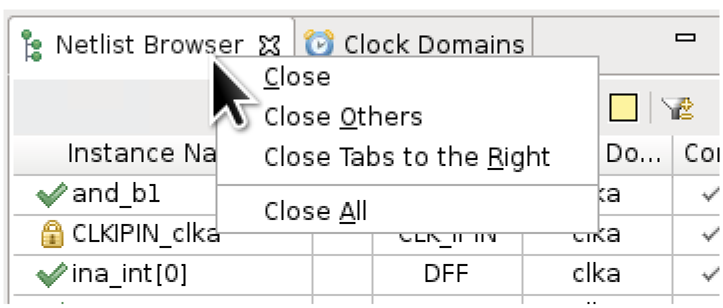


Figure 4: ***Example of View Right-click Context Menu***

Some views have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view.

Some views also have their own menus to affect the content of the view. When such a menu is available, a vertical elipsis (three stacked dots:) appear at the far right of the view toolbar. To open the menu for a view, click the icon.

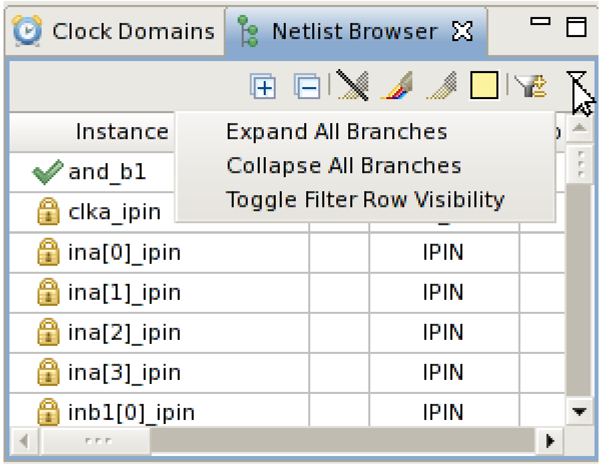


Figure 5: ***Example of View Toolbar Including Opened View Menu***

Views are typically grouped by shared context into Perspectives (see page 24). Within a perspective, a view might appear by itself, or stacked with other views in a tabbed notebook. The layout of a perspective can be changed by opening and closing views and by docking them in different positions in the Workbench (see page 24) window. See Working with Views and Editors (see page 262) for more information.

The views contained by the Project Perspective are:

- Projects View (see page 125)
- Flow View (see page 61)
- Options View (see page 103)
- Tcl Console View (see page 144)

The Multiprocess View (see page 83) is also part of this perspective, but is hidden by default.

The views within the Floorplanner Perspective are:

- Search View (see page 132)
- Selection View (see page 136)
- Critical Paths View (see page 48)
- Critical Path Diagram View (see page 45)
- Floorplanner View (see page 53)
- Clock Regions View (see page 36)
- Clock Domains View (see page 33)
- Placement Regions View (see page 120)
- Partitions View (see page 117)
- Netlist Browser View (see page 89)
- Properties View (see page 128)
- Tcl Console View (see page 144)

The views contained within the IP Configuration Perspective are:

- Outline View (see page 112)

- IP Libraries View (see page 81)
- IP Diagram View (see page 80)
- IP Problems View (see page 81)
- Tcl Console View (see page 144)

The views of the Programming and Debug Perspective are:

- Snapshot Debugger View (see page 139)
- Download View (see page 51)
- Tcl Console View (see page 144)

The views of the HW Demo Perspective are:

- HW Demo View (see page 64)
- Snapshot Debugger View (see page 139)
- Tcl Console View (see page 144)

Clock Domains View

The Clock Domains view provides a table listing all the clock domains found in the active design. Counts are also provided of the major logic types within each clock domain. Similar to the Netlist Browser View (see page 89), filters are available for each column of the table, so that in cases where there are many clock domains in a design, the visible content of the table may be limited to just those clock domains meeting the chosen filter criteria. Filters are available for all columns of the table except the Highlight Color column.

By default, the Clock Domains view is included in the Floorplanner perspective (see page 24). To add it to other perspectives, select **Window** → **Show View...** → **Other...** → **Achronix** → **Clock Domains**.

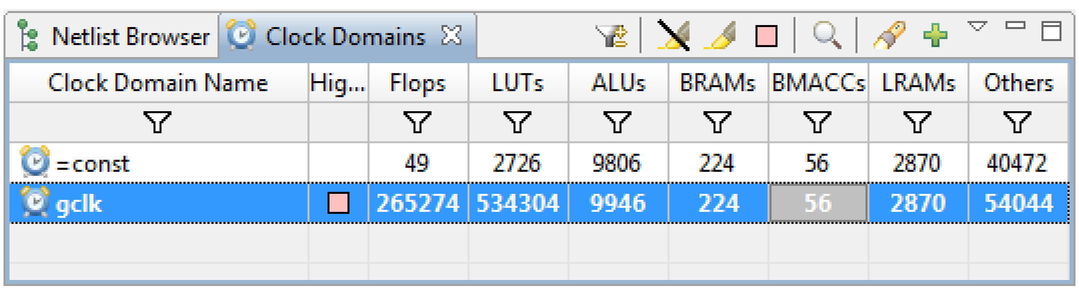


Figure 6: ***Clock Domains View Example***

The various Achronix target devices contain different mixes of the possible resource types. Accordingly, the resource type columns in this view (i.e., flops, BRAMs, ALUs, etc.) are dynamic, and change to match the target device after the Prepare flow step has been run. The screenshots and example descriptions in this document section might not exactly reflect the current target device.

Table 5: *Clock Domains View Columns*

Column Name	Description
Clock Domain Name	The name of the clock domain in the active design.
Highlight Color	If all instances within the clock domain have the same highlight color, the row shows a color square with that same highlight color. If even one contained instance has a differing highlight color, or no highlight at all, then the row displays no color square.
Resource	A different column is provided for each *resource* type within the target device. Each table cell in that column shows the sum count of all contained *resource* instances within the named clock domain for that row.

A number of actions are available in the view, via buttons at the top of the view and (right-click) context menus on the rows of the table.

Table 6: *Clock Domains View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	Add Instances to Selection	Y	Y	Adds the instances within the clock domain to the ACE Selection Set (as shown in the Selection View (see page 136)).
	Choose Highlight Color	Y		Determines which color is applied to the objects chosen the next time the Highlight action is selected for this view.
	Highlight	Y	Y	Applies the currently active Highlight color to the instances within the chosen clock domain (see Highlighting Objects in the Floorplanner View (see page 315)).
	Un-Highlight	Y	Y	Clears the Highlight for the instances within the chosen clock domain.
	Zoom To	Y	Y	Zooms the Floorplanner View (see page 53) to a region containing the instances within the clock domain currently chosen in the tree.
	Search for Instances	Y	Y	Searches for instances belonging to the chosen clock domain. A Tcl `find` command is issued, and the Search View (see page 132) is populated with the results.
	Toggle Filter Row Visibility [1]	Y		Changes whether the filter row (of filter icons) is visible or not.

Table Notes

1. Toggle Filter Row Visibility does not alter whether filters are active, it only changes filter icon visibility.

Organizing Table Data

The following are ways to alter the presentation of the data in the Clock Domains table:

Column Resizing

The width of a column can be changed as follows:

1. place the mouse cursor over the boundary between columns (the mouse cursor changes to indicate resizing is possible).
2. Simply left-click and drag left or right to resize the column to the desired width, then release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold on any column name
2. Drag left or right to move the column between any other pair of columns
3. Release the left mouse button to insert the column header at the new location

While dragging, the dragged column header appears alongside the mouse cursor, plus a thick column header separator showing where the column insertion occurs when the mouse is released.

Filtering

Most columns of the table may filter the displayed clock domains by value. When filtering by column value, only clock domains with column values matching the filter are retained; non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the Overlay Color column) lack a filter icon in the filter row.

To add a filter to a column:

1. The Filter Row must first be visible. Select the ( ) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the ( ) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the ( ) active filter icon.

To edit (or clear) an existing filter:

1. Click the ( ) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.

3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the ( ) inactive version.

Drag-and-Drop

The Clock Domains view supports a limited set of Drag-and-Drop interactions with other views in the Floorplanner perspective (see page 24). The view only acts as a Drag-and-Drop source; items dropped on the Clock Domains view will be ignored.

Any row of the table may be dragged to the Tcl Console view (see page 144), and when dropped anywhere in the view the clock domain name (with the appropriate object type prefix) is inserted at the beginning of the Tcl command-line.

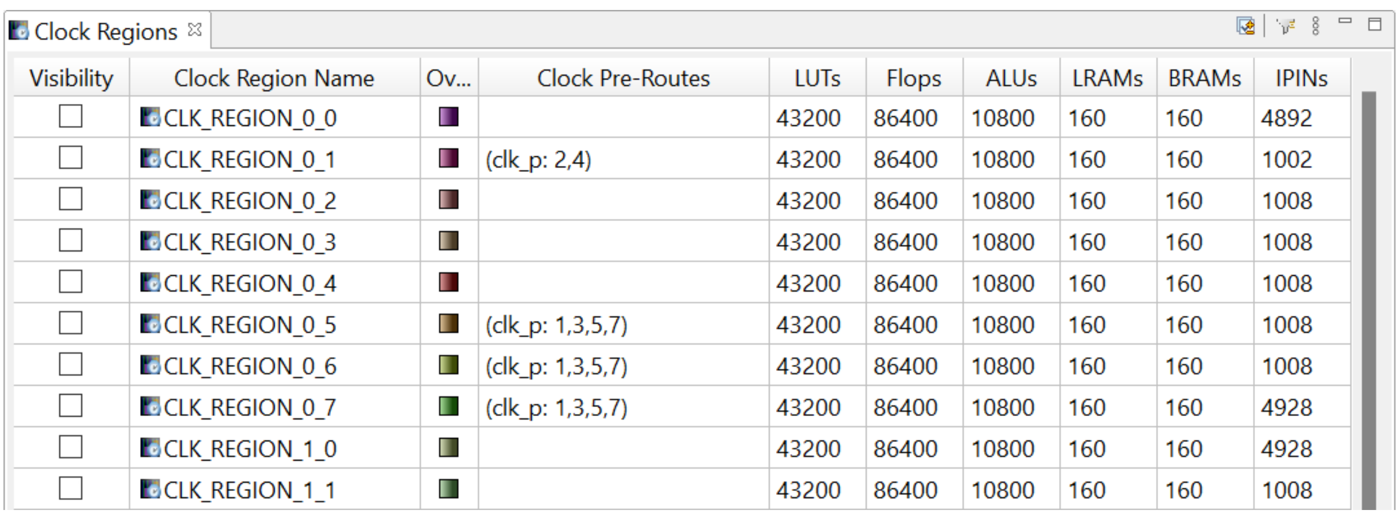
Any clock domain in the table may be dragged to the Placement Regions view (see page 120) or the Floorplanner View (see page 53) (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 358). Dragging a clock domain is the equivalent of dragging all individual instances which are members of that clock domain. Be aware that since placement regions may only encompass the fabric core and boundary region, but not the I/O ring, any dragged I/O instances are not assigned to placement regions.

Save

Clock Regions View

The Clock Regions view provides a tabular representation of the site type content of each of the Clock Regions (see page 243) in the currently selected Target Device, and allows toggling the visibility of the overlay within the Floorplanner View (see page 53) for each Clock Region. The view table remains empty until the currently active Implementation has been prepared (i.e., the **Run Prepare** flow step has been completed).

By default, the Clock Regions view is included in the Floorplanner perspective (see page 24). To add the view to the current perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Clock Regions**.



Visibility	Clock Region Name	Ov...	Clock Pre-Routes	LUTs	Flops	ALUs	LRAMs	BRAMs	IPINs
	CLK_REGION_0_0			43200	86400	10800	160	160	4892
	CLK_REGION_0_1		(clk_p: 2,4)	43200	86400	10800	160	160	1002
	CLK_REGION_0_2			43200	86400	10800	160	160	1008
	CLK_REGION_0_3			43200	86400	10800	160	160	1008
	CLK_REGION_0_4			43200	86400	10800	160	160	1008
	CLK_REGION_0_5		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	1008
	CLK_REGION_0_6		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	1008
	CLK_REGION_0_7		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	4928
	CLK_REGION_1_0			43200	86400	10800	160	160	4928
	CLK_REGION_1_1			43200	86400	10800	160	160	1008

Figure 7: ***Clock Regions View Example***

Caution!

Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The resource type columns shown in the screenshot are examples only, and do not match all target devices.

Table 7: *Clock Regions Table Columns*

Column	Editable	Description
Visibility	Y	When checked, the clock region overlay is painted in the Floorplanner View (see page 53), using the translucent color shown in the **Overlay Color** column.
Name		The name of this clock region.
Overlay Color	Y	The color used to paint the location of the clock region as an overlay in the Floorplanner View. Right-click any row, then choose **Change Overlay Color** to choose an alternate overlay paint color for that clock region.
Clock Pre-Routes		If any clock pre-routes exist for a given partition, they are listed here.
Resource		The number of *resource* sites contained in this clock region.

Table 8: ***Clock Regions View Actions***

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show / Hide overlay		Y		Show or Hide the overlay for the chosen clock region in the Floorplanner View.
	Change Overlay Color		Y		Allows changing the translucent overlay color which is used to paint the chosen clock region in the Floorplanner View (when the visibility is enabled).
	Reset Overlay Color		Y		Reset the chosen clock region overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two clock regions are too similar for easy discernment, this action pseudo-randomly picks another color. Each time this action is chosen, another color is picked.
	Zoom To		Y		Pans and zooms the Floorplanner View to show the location of the selected clock region. (1)
	Reset All Overlay Colors			Y	Pseudo-randomly reassigns new overlay colors for all clock regions.
	Configure Clock Pre-Routes...		Y		Allows adding clock pre-route information for the selected partition(s).
	Show / Hide All Clock Regions	Y	Y	Y	Toggles the visibility checkboxes for all clock domains in the table, causing all to be alternately shown or hidden in the Floorplanner View.
	Toggle Filter Row Visibility	Y	Y	Y	Changes whether the filter row (of filter icons) at the top of the table is visible or not. (2)

Table Notes

1. If the clock region visibility column checkbox is disabled, the clock region overlay is not painted and thus is not visible.
2. This toggle does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Using the Table to Display Clock Regions in the Floorplanner View

Each clock region is automatically given a unique translucent overlay color to represent the clock region when painting the Floorplanner View (see page 53). By default, no clock region overlays are painted in the Floorplanner View. Clock region overlays must be enabled if they are to be displayed. The overlay color may optionally be altered for each/all clock regions, but these color choices do not persist between ACE sessions.

> **Note**
>
> While alternate overlay colors are allowed to be chosen for each clock region, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each clock region.

The following are ways to alter the presentation of Clock Region data in the Floorplanner View (see page 53):

Enable/Disable Painting of Individual Clock Regions Within the Target Device:

When the checkbox in the **Visibility** column for a clock region is selected, the area of the target device (in the Floorplanner view) representing that clock region is painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view is redrawn with the chosen clock region overlay no longer painted.

Enable/Disable Painting of all Clock Regions Within the Target Device:

When the **Show/Hide All Clock Regions** action is selected, the visibility of all clock regions are simultaneously either enabled or disabled, causing the Floorplanner View to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Clock Regions:

The overlay rendering color of each individual clock region may be chosen as follows:

1. Right-click the mouse pointer anywhere on the row of the desired clock region.
2. Select **Choose Overlay Color** from the popup context menu.
3. The Color Dialog may then be used to choose the desired color for the clock region.

> **Note**
>
> This is a temporary color change — colors are reverted to automatically chosen defaults if changes are made to the active design, the active implementation, the target device, or ACE is closed.

ACE automatically picks a different overlay color for an individual clock region if **Reset Overlay Color** is selected from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Clock Regions:

ACE automatically picks different overlay colors for all clock regions if the **Reset All Overlay Colors** action is selected from the Clock Regions View local pull-down menu.

Organizing Table Data

The following are ways to alter the presentation of the data in the Clock Regions table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns — at this point the mouse cursor should change to indicate resizing is possible.
2. Simply left-click and drag left or right to resize the column to the desired width.
3. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold on any column name
2. Drag left or right to move the column between any other pair of columns.
3. Release the left mouse button to insert the column header at the new location.

While dragging, the dragged column header is visible alongside the mouse cursor, and there is a thick column header separator showing where the column insertion occurs if the mouse is released at the present cursor location.

Filtering

Most columns of the table may filter the displayed clock regions by value. When filtering by column value, only clock regions with column values matching the filter are retained; non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the Overlay Color column) lack a filter icon in the filter row.

To add a filter to a column:

1. The Filter Row must first be visible. Select the ( ) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the ( ) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the ( ) active filter icon.

To edit (or clear) an existing filter:

1. Click the ( ) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the ( ) inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all clock regions marked as "visible" in the view. Making more or fewer clock regions visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Clusters View

The Clusters view allows the toggling the visibility of overlays within the Floorplanner View (see page 53) for each logic cluster. The view table remains empty until the currently active Implementation has been prepared (i.e. the **Run Prepare** flow step has been completed).

By default, the Clusters view is included in the Floorplanner perspective (see page 24). To add the view to the current perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Clusters**.

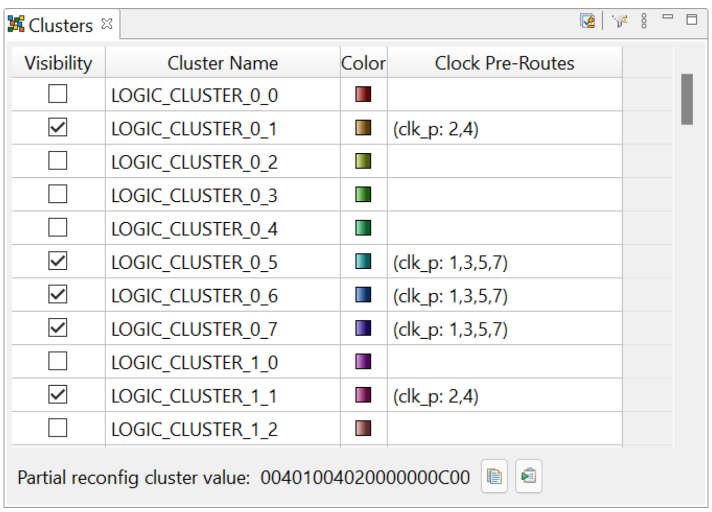


Figure 8: *Clusters View Example*

Table 9: *Clusters Table Columns*

Column	Editable	Description
Visibility	Y	When checked, the logic cluster overlay is painted in the Floorplanner View (see page 53), using the translucent color shown in the **Color** column.
Cluster Name		The name of this cluster.
Color	Y	The color used to paint the location of the cluster as an overlay in the Floorplanner View. Right-click any row, then choose **Change Overlay Color** to choose an alternate overlay paint color for that cluster.
Clock Pre-Routes		If any clock pre-routes exist for a given partition, they are listed here.

Table 10: *Clusters View Actions*

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show / Hide overlay		Y		Show or Hide the overlay for the chosen cluster in the Floorplanner View.
	Change Overlay Color		Y		Allows changing the translucent overlay color which is used to paint the chosen cluster in the Floorplanner View (when the visibility is enabled).
	Reset Overlay Color		Y		Reset the chosen cluster overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two clusters are too similar for easy discernment, another color is pseudo-randomly selected. Each time this action is chosen, another color is selected.
	Zoom To		Y		Pans and zooms the Floorplanner View to show the location of the selected cluster. (1)
	Reset All Overlay Colors			Y	Pseudo-randomly reassigns new overlay colors for all clusters.
	Configure Clock Pre-Routes...		Y		Allows adding clock pre-route information for the selected partition(s).
	Show /Hide All Custers	Y	Y	Y	This toggles the visibility checkboxes for all logic clusters in the table, causing all to be alternately shown or hidden in the Floorplanner View.
	Toggle Filter Row Visibility	Y	Y	Y	Changes whether the filter row (of filter icons) at the top of the table is visible or not. (2)

Table Notes

1. If the cluster visibility column checkbox is disabled, the cluster overlay is not painted and thus is not visible.
2. This toggle does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Using the Table to Display Clusters in the Floorplanner View

Each cluster is automatically given a unique translucent overlay color to represent the cluster when painting the Floorplanner View (see page 53). By default, no cluster overlays are painted in the Floorplanner View. The cluster overlays must be enabled to be displayed. The overlay color for each/all clusters may optionally be altered, but these color choices are not persisted between ACE sessions.

> While choosing alternate overlay colors is allowed for each cluster, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each cluster.

The following are ways to alter the presentation of Cluster data in the Floorplanner View (see page 53):

Enable/Disable Painting of Individual Clusters Within the Target Device:

When the checkbox in the **Visibility** column for a cluster is selected, the area of the target device (in the Floorplanner view) representing that cluster is painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view is redrawn with the chosen cluster overlay no longer painted.

Enable/Disable Painting of All Clusters Within the Target Device:

When the **Show/Hide All Clusters** action is chosen, the visibility of all clusters are simultaneously either enabled or disabled, causing the Floorplanner View to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Clusters:

The overlay rendering color of each individual cluster may be chosen as follows:

1. Right-click the mouse pointer anywhere on the row of the desired cluster.
2. Select **Choose Overlay Color** from the popup context menu.
3. The Color Dialog can then be used to choose the desired color for the cluster.

> **Note**
>
> This is a temporary color change — colors are reverted to automatically chosen defaults if changes are made to the active design, the active implementation, the target device, or ACE is closed.

ACE automatically picks a different overlay color for an individual cluster if **Reset Overlay Color** is chosen from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Clusters:

ACE automatically picks different overlay colors for all clusters if the **Reset All Overlay Colors** action is chosen from the clusters View local pull-down menu.

Organizing Table Data

The following are ways to alter the presentation of the data in the clusters table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns.

2. At this point, the mouse cursor should change to indicate resizing is possible.
3. Simply left-click and drag left or right to resize the column to the desired width.
4. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold on any column name.
2. Drag left or right to move the column between any other pair of columns.
3. Release the left mouse button to insert the column header at the new location.

While dragging, the dragged column header is visible alongside the mouse cursor and there is a thick column header separator showing where the column insertion is to occur if the mouse is released at the present cursor location.

Filtering

Most columns of the table may filter the displayed clusters by value. When filtering by column value, only clusters with column values matching the filter are retained. Non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the Overlay Color column) lack a filter icon in the filter row.

To add a filter to a column:

1. The Filter Row must first be visible. Select the ( ) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the ( ) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the ( ) active filter icon.

To edit (or clear) an existing filter:

1. Click the ( ) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the ( ) inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all clusters marked as "visible" in the view. Making more or fewer clusters visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Critical Path Diagram View

The Critical Path Diagram view provides a graphical representation of a single critical path. Selecting a row in the Critical Paths View (see page 48) table updates the Critical Path Diagram view diagram so that it contains a graphical representation of the selected critical path. The graphical representations consist of circular nodes (representing instances) connected by arrows (representing one or more nets). Similar to the Floorplanner View (see page 53), the Critical Path Diagram view contains a Fly-out Palette of display options on the right, and a collection of buttons at the top. The colors used in the Critical Path Diagram view are configured via the Critical Path Diagram View Preference Page (see page 188). For details on usage of the diagram view, please see Using Critical Path Diagrams (see page 327) or Analyzing Critical Paths (see page 324).

By default, the Critical Path Diagram view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window → Show View... → Critical Path Diagram**.

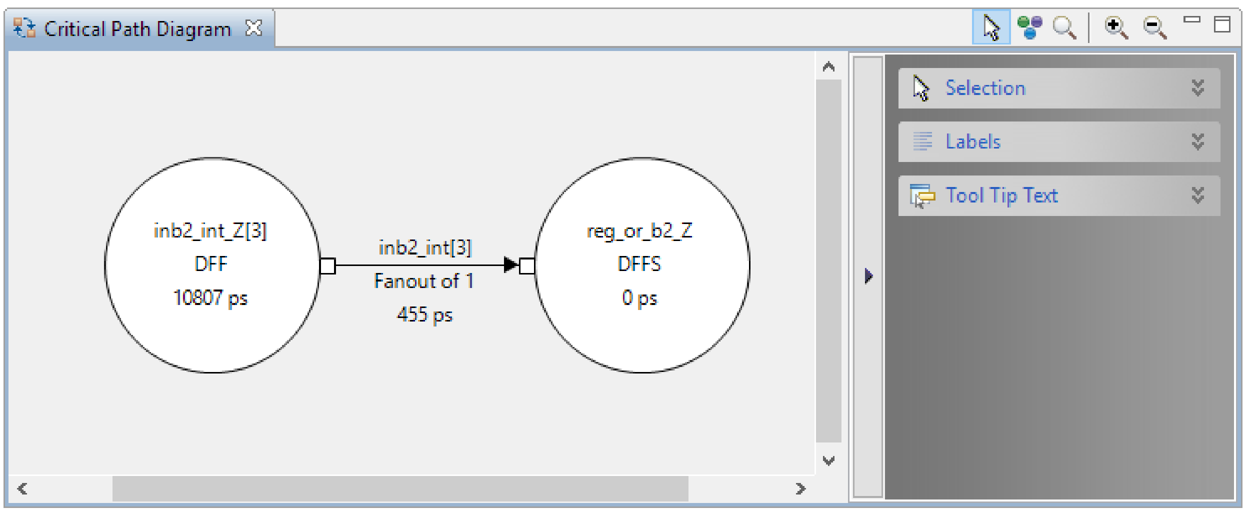


Table 11: *Critical Path Diagram View Example*

When no critical path is selected in the Critical Paths View (see page 48), the diagram displays a warning.

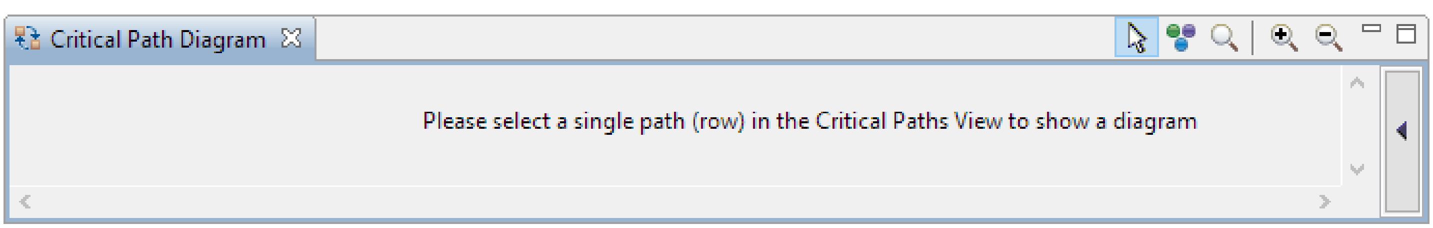


Figure 9: *No Critical Path Selected Warning*

Table 12: *Critical Path Diagram View Toolbar Buttons*

Icon	Action	Description
	Selection tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The selection tool creates a selection rectangle when dragging with the left mouse button. Any objects in the selection rectangle are either added to or removed from the current ACE selection set, as configured in the fly-out palette.
	Movement tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The movement tool pans the view when dragging with the left mouse button.
	Zoom tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The zoom tool creates a zoom-in rectangle when the left mouse button is pressed and held, then dragged to the lower-right. The zoom tool creates a zoom-out line when the left mouse button is pressed and held, then dragged to the upper-left.
	Zoom in	Increases the current zoom level in the Critical Path Diagram view by 200%.
	Zoom out	Decreases the current zoom level in the Critical Path Diagram view by 200%.

Right-clicking on an instance or net within the diagram also displays a context menu of additional actions.

Table 13: *Critical Path Diagram View Context Menu Actions*

Icon	Action	Description
	Add to Selection	The instance or net under the mouse is added to the ACE selection set (and painted in the selection color).
	Remove from Selection	The instance or net under the mouse is removed from the ACE selection set if currently selected (and thus no longer is painted the selection color).
	Highlight	Sets the highlight color for the instance or net under the mouse to the currently-chosen view highlight color.
	Choose highlight color	Determines which color is applied to instances/nets the next time the Highlight action is selected for this view.
	Un-Highlight	Turns off the highlight color for the instance or net under the mouse.
	Zoom To	Pans and zooms the Floorplanner view (not this view) to the closest zoom that still displays (centered) the entire instance or net currently under the mouse in the Critical Path Diagram.
	Show in Netlist (1)	Attempts to open a text editor to the file and line number relevant to the instance or net under the mouse.
	Fix Placement of Instance	Causes the placement state of the Instance under the mouse cursor to change from unfixed (or soft) to fixed.

Icon	Action	Description
	Unfix Placement of Instance	Causes the placement state of the Instance under the mouse cursor to change from fixed to unfixed (or soft).
	Unplace Instance	Causes the placed instance under the mouse cursor to be unplaced, vacating the site.
	Unplace All Selected Instances	Causes all Instances currently in the ACE Selection Set (as listed in the Selection View (see page 136)) to be unplaced at once (this is much more efficient than unplacing multiple instances individually).

Table Notes

1. This Early Access functionality might not always open the text editor to the expected location.

Fly-Out Palette

The following options are available in the fly-out palette in the Critical Path Diagram view.

Selection

The ( ) Selection Options control the selection of objects in the Critical Path Diagram view.

Table 14: *Selection Options*

Option	Default	Description
Select	Enabled	This radio button controls the action applied to objects in the selection region. This setting causes the objects to be added to the current ACE selection set.
Deselect	Disabled	This radio button controls the action applied to objects in the selection region. This setting causes the objects to be removed from the current ACE selection set.

Label

The ( ) Label options control the text labels on graph nodes and arrows (instances and net abstractions) in the Critical Path Diagram view. Note that these labels are only displayed when there is enough room for them to be printed on-screen. It might be necessary to **Zoom In** to provide sufficient area for all the desired text to be displayed.

Table 15: *Label Options*

Option	Description
Instance Names	Displays the instance name each graph node represents.
Instance Types	Displays the instance type (cell) for each graph node.
Net Names	Displays the net name represented by each arrow.
Delays	Displays the delay (in ps) to traverse each node or arrow.
Fanouts	Displays the fanout of the net represented by the arrow.

Tool Tip Text

The ( ) Tooltip options control the tooltip content while hovering over graph nodes and arrows in the Critical Path Diagram view.

Table 16: *Tooltip Options*

Option	Description
None	No tooltips are displayed for nodes or arrows.
Instance Names	Displays the instance name each graph node represents.
Instance Types	Displays the instance type (cell) for each graph node.
Net Names	Displays the net name represented by each arrow.
Pin Names	Displays source and sink pin names for nets.
Delays	Displays the delay (in ps) to traverse each node or arrow.
Fanouts	Displays the fanout of the net represented by the arrow.

Critical Paths View

The Critical Paths view provides a table of critical paths resulting from running timing analysis. This view (in cooperation with the Critical Path Diagram View (see page 45)) displays critical path details, manages selection of objects on critical paths, and highlights critical paths in the Floorplanner View (see page 53).

Clicking a row in the table enables toolbar buttons for analyzing the associated critical path, and causes the display of a graphical diagram of the associated critical path in the Critical Path Diagram View (see page 45). Clicking a column header sorts the table according to the data in that column.

By default, the Critical Paths view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window** → **Show View...** → **Critical Paths**.

Table 17: *Critical Path View Actions*

Icon	Action	Description
	Select path	Adds the selected critical path in the table to the current ACE selection set.
	Select pins	Adds pins on the selected critical path in the table to the current ACE selection set.
	Select instances	Adds instances on the selected critical path in the table to the current ACE selection set.
	Select nets	Adds nets on the selected critical path in the table to the current ACE selection set.
	Zoom to path	Zooms the Floorplanner view to a region containing the selected critical path in the table.
	Print Path Details	Produces a detailed report of the selected critical path in the table to the text output in the Tcl Console view.
	Save Script File	Displays the Save Script File dialog which allows saving a Tcl script of find commands for use in the schematic viewer of the synthesis tool.

The view is primarily made up of a tree table, with each branch of the tree representing a separate clock domain. The most critical path of each clock domain is the branch node, with all other paths from that clock domain acting as leaves for that branch. Setup violations are considered "worse" than hold violations, thus any setup violation takes precedence over hold violations as the branch node, regardless of relative slack values.

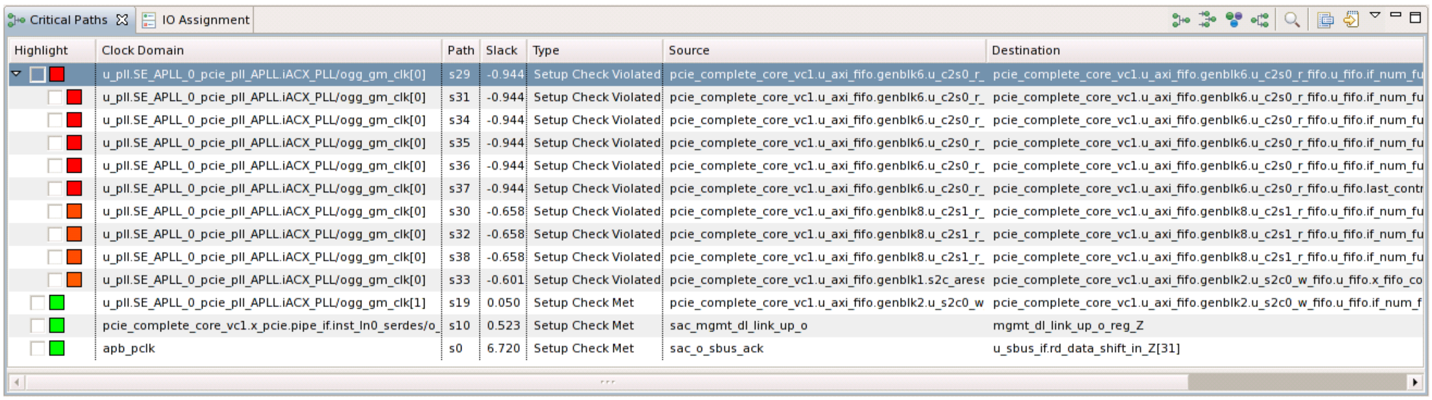


Figure 10: *Critical Path View Example*

Entries in the table are always grouped by clock domain, with individual paths sorted within a clock domain. The clock domains themselves are (by default) sorted from most critical to least critical.

The default sort order, from most critical to least critical, of the critical paths is as follows:

1. Setup violations, from the most negative slack value to zero.
2. Hold violations, from the most negative slack value to zero.
3. Setup met, from zero to the most positive slack.

4. Hold met, from zero to the most positive slack.

Table 18: *Critical Path View Table Columns*

Column	Description
Highlight	Allows highlighting the path in the Floorplanner view, using the checkbox. Also allows configuring the highlight color of the path by clicking the color selector box.
Clock Domain	Displays the clock domain name of the path.
Path	Displays the unique path ID (used in the Timing Report (see page 227)).
Slack	Displays the slack for the path in ns.
Type	Displays the path type.
Source	Displays the source instance of the path.
Destination	Displays the destination instance of the path.

By default, the highlight colors for the Setup Violation and Hold Violation path types ranges from red (the worst slack values) through orange to yellow (any Violation slack values close to zero). The default highlight color of Slack Met and Hold Met path types is always green, and does not vary by reported slack value.

The View local pull-down menu (found to the right of the View Toolbar buttons) contains some additional controls for the view: four filters to control which Types of paths are displayed in the table, as well as shortcuts to run the four stages of timing analysis. As mentioned previously, the most critical path within each clock domain is always displayed, regardless of the type filter settings. (Every clock domain is always represented in the tree table by at least one row of data.)

Table 19: *Critical Paths View Drop-down Menu Actions*

Action	Description
Show Clock Paths	If checked, highlighted critical paths in the Floorplanner View show the clock routing segments as part of the critical path. If unchecked, only the data portion of the critical path is shown.
Show Setup Violations	If checked, Setup Violation leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Hold Violations	If checked, Hold Violation leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Setup Met	If checked, Setup Met leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Hold Met	If checked, Hold Met leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Run Prepared Timing Analysis	If selected, runs the Prepared Timing Analysis flow step.

Action	Description
Run Post-Place Timing Analysis	If selected, runs the Post-Place Timing Analysis flow step.
Run Post-Route Timing Analysis	If selected, runs the Post-Route Timing Analysis flow step.
Run Final Timing Analysis	If selected, runs the Final Timing Analysis flow step.

Download View

The Download view provides a graphical interface for playing a STAPL file to an Achronix FPGA connected via a Bitporter pod or FTDI FT2232H device. By default, the Download view is included in the Programming and Debug Perspective (see page 24). To access the Download view, select **Window** → **Show View...** → **Others** → **Download View**.

When the Download view opens, the windows might need to be resized for optimal viewing.

Caution!

- **The JTAG connection must be configured before using the Download View!**
  ACE interacts with the FPGA using the JTAG interface through a Bitporter pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the Download view. The configuration is managed using the Configure JTAG Connection Preference Page (see page 186), which is easily accessible by pressing the ( ) **Configure JTAG Interface** button in the Download view. See Configuring the JTAG Connection (see page 331) for more details.
- While using the Download view, it is strongly recommended that the Tcl Console View (see page 144) be kept visible to display any status or error messages returned from the external `acx_stapl_player` process.

See also: Playing a STAPL File (Programming a Device) (see page 350).

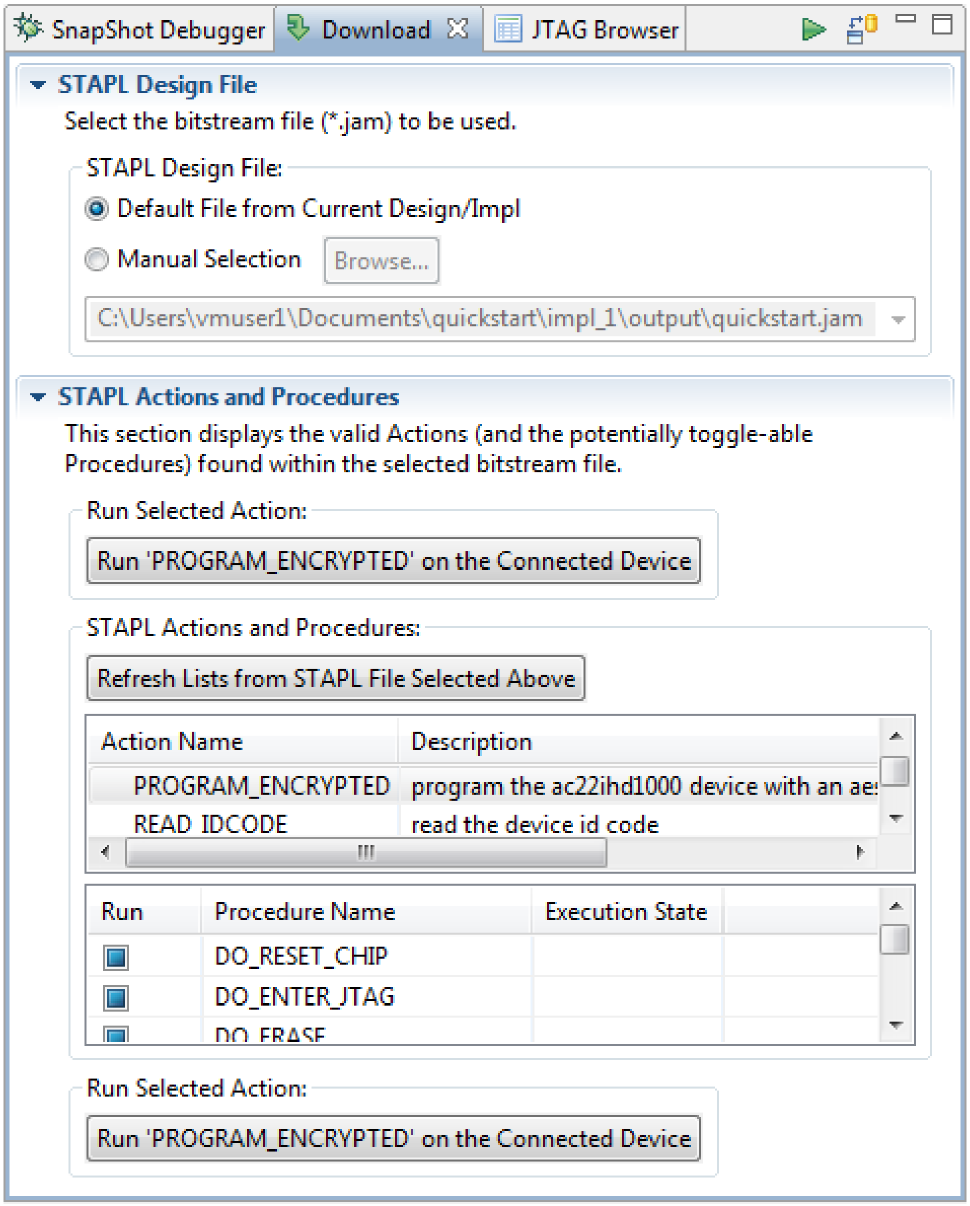


Figure 11: ***Download View Example***

Table 20: *Download View Options*

Option	Description
STAPL Design File	
Default File from Current Design/Impl	The STAPL design file corresponds to the bitstream file of the currently active implementation (`[design_name].jam`), as generated during the **Generate Bitstream**Flow Step (see page 221).
Manual Selection and **Browse**	Allow choosing or entering any STAPL design file from the file system. The textfield showing the filename is also a combo-box of the last fifteen `*.jam` files selected.
STAPL Actions and Procedures	
Refresh lists from STAPL File Selected Above	Button to refresh the lists of actions and procedures to match those found in the selected STAPL file.
Action Name	Lists the Actions contained in the selected STAPL file.
Description	Lists the description for each Action found in the file.
Run	Allows the named Procedure to be run or bypassed. Icons indicate: = Required (always run; cannot be deselected) = Deselected (not run; may be toggled) = Selected (run; may be toggled)
Procedure Name	Displays the names of the Procedures comprising the selected Action, in execution order.
Execution State	Displays the execution state of a Procedure. Values include: • blank (required) • optional (disabled by default, but may be enabled) • recommended (enabled by default, but may be disabled)
Run Selected Action	Using the selected active Procedures, plays the selected Action to the connected Achronix FPGA. Equivalent to Tcl command `run_stapl_action` .

Floorplanner View

The Floorplanner view provides a graphical view of the physical layout of the device. This view allows visualizing the device, place and route data, critical paths, and the current selection set. The view allows zooming out to see a general overview of the user design mapped onto the device, or zooming in to see specific details.

Clicking the tall narrow arrow button on the far right of the Floorplanner view shows or hides the Fly-Out Palette (see page 56) of display options.

By default, the Floorplanner view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window** → **Show View** → **Other...** → **Floorplanner**.

See also:

- Viewing the Floorplanner (see page 312)
- Pre-placing a design (see page 318)
- Floorplanner View Colors and Layers Preference Page (see page 190)
- Floorplanner View Optimizations Preference Page (see page 195).

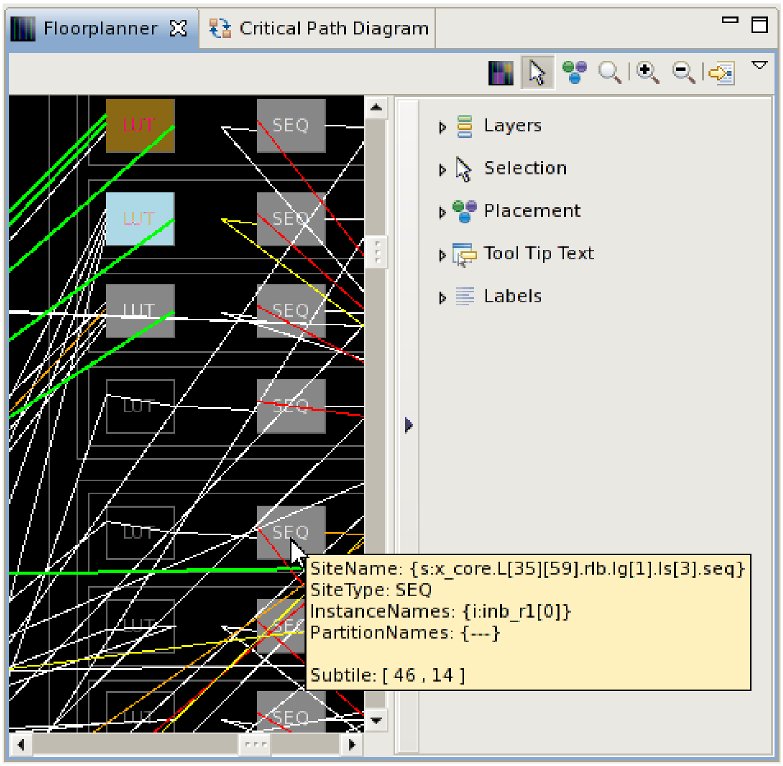
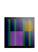


Figure 12: *Floorplanner View Example, Including Expanded Fly-Out Palette*

Table 21: *Floorplanner View Toolbar Buttons*

Icon	Action	Description
	Placement Region tool	Controls the behavior of the mouse while in the Floorplanner view. Allows the manipulation of Placement Regions and Placement Region Constraints (see page 354). When the Placement Region Tool is active, the mouse may be used to create new placement regions (see page 355), move (see page 357) or resize (see page 357) existing placement regions, and/or assign objects to placement region constraints (see page 358).
	Selection tool	Controls the behavior of the mouse while in the Floorplanner view. The selection tool creates a selection rectangle when the left mouse button is pressed and held. Any objects in the selection rectangle are applied with the current selection action, as configured in the fly-out palette.
	Placement tool	Controls the behavior of the mouse while in the Floorplanner view. The placement tool either pans the view or allows drag-and-drop placement of instances with the mouse drag when the left mouse button is pressed and held.

Icon	Action	Description
	Zoom tool	Controls the behavior of the mouse while in the Floorplanner view. The zoom tool creates a zoom-in rectangle when the left mouse button is pressed and held, then dragged to the lower-right. The zoom tool creates a zoom-out line when the left mouse button is pressed and held, then dragged to the upper-left.
	Zoom in	Increases the current zoom level in the Floorplanner view by 200%.
	Zoom out	Decreases the current zoom level in the Floorplanner view by 200%.
	Save Pre-placement Constraints	Opens the Save Placement dialog (see page 170) allowing the current placement to be saved to a pre-placement constraints file (`.pdc`).

An Instance or Net within the Floorplanner may also be right-clicked to display a context menu of additional actions.

Table 22: *Floorplanner View Context Menu Actions*

Icon	Action	Description
	Add to Selection	The Instance or Net under the mouse is added to the ACE selection set (and is painted in the Selection color).
	Remove from Selection	The Instance or Net under the mouse is removed from the ACE selection set if currently Selected (and thus is no longer painted the Selection color).
	Highlight	Sets the highlight color for the Instance or Net under the mouse to the currently-chosen Floorplanner view highlight color.
	Choose highlight color	Determines which color is applied to Instances/Nets the next time the Highlight action is selected for this view.
	Un-Highlight	Turns off the highlight color for the Instance or Net under the mouse.
	Zoom To	Pans and zooms the Floorplanner view to the closest zoom that still displays (centered) the entire Instance or Net under the mouse.
	Show in Netlist (1)	Attempts to open a text editor to the file and line number relevant to the Instance or Net under the mouse.
	Fix Placement of Instance	Causes the placement state of the Instance under the mouse cursor to change from unfixed (or soft) to Fixed.
	Unfix Placement of Instance	Causes the placement state of the Instance under the mouse cursor to change from Fixed to unfixed (or soft).

Icon	Action	Description
	Unplace Instance	Causes the placed instance under the mouse cursor to be unplaced, vacating the site.
	Unplace All Selected Instances	Causes all Instances currently in the ACE Selection Set (as listed in the Selection View (see page 136)) to be unplaced at once (this is much more efficient than unplacing multiple instances individually).

Table Notes

1. "Show in Netlist" is early access functionality and might not always open the text editor to the expected location.

Panning and Zooming

The Floorplanner view allows zooming in and out, to see more or less details respectively. There are several ways to change the zoom level:

1. Use the mouse scroll wheel.
2. Use the ( ) **Zoom In** and ( ) **Zoom Out** buttons in the toolbar.
3. Use keyboard shortcuts (most frequently used).

See the task Zooming the Floorplanner In and Out (see page 312) for complete details.

Most of the other views within the Floorplanner Perspective also include context-sensitive Actions to **Zoom To** chosen individual objects or groups of objects — these actions cause the Floorplanner to center the chosen object(s) in the Floorplanner, and to change the zoom level so that the chosen object(s) are as large/detailed as possible without overflowing the visible area.

When zoomed in, the FPGA requires more area than can easily fit in the view, making it necessary to pan the view around to see the different areas of the FPGA. Panning is most frequently performed using the arrow keys on the keyboard, mouse interactions with the scrollbars, or the Placement Tool drag-and-drop interactions. See the task Floorplanner Panning (see page 313) for complete details.

Note

When painting objects in the Floorplanner, when the view is zoomed out, some objects become too small to be rendered with any detail. These objects are painted, at a minimum, as a single pixel of the appropriate color.

Empty Sites (those without a placed instance) are a special case. Unless Selected, sites that are too small are not painted at all, even if layer settings would otherwise allow them to be visible. Selected sites are always painted, with a minimum size of a single pixel.

When a single pixel represents multiple objects, as happens when zoomed all the way out, ACE paints only the most critical or most important object state at that pixel location, so the single pixel is the most critical or most important color. The relative priorities of the states are described in Instance States (see page 243).

Fly-Out Palette

The following options are available in the fly-out palette in the Floorplanner view:

Layers

The ( ) Layer Options control several layers of visible data in the Floorplanner view, allowing filtering the view so it contains a desired subset of all the available information.

Table 23: *Layer Options*

Option	Default	Description
Instances	Enabled	This layer shows all placed instances.
Selected Instance Flylines (1)	Disabled	This layer shows flylines representing the net connections of selected instances (instances in the ACE selection set).
Clock Nets	Enabled	This layer shows all clock nets.
Non-clock Nets	Enabled	This layer shows all non-clock nets.
Routing Status		
Open Connections (2)	Disabled	Toggles the display of Open portions of a net. Open Connections are displayed in the same color as the routed portion of a net, but with a dotted line instead of a solid line. Open connections are a subset of a normal net, and are thus also managed by the layer options for **Non-clock Routes**, **Clock Routes**, and **Route Drawing Mode**.
Open Pins	Disabled	Toggles the display of squares (red by default) highlighting the pins at the endpoints of Open Connections.
Overflows	Disabled	Toggles the display of diamonds (orange by default) highlighting pins where route overflows occur. (This is very rare.)
Pins	Disabled	This layer shows all pins on each Site
Sites	Enabled	This layer shows all the Sites on the device.

Table Notes

1. The displayed flylines are filtered by the **Non-clock Routes** and **Clock Routes** layer checkboxes. If only **Clock Routes** is checked, then only the flylines for clock nets of the selected instance(s) are displayed.
2. **Caution:** Dotted lines, as used for Open Connections, are much slower to render than solid lines. Thus, it is recommended that Open Connections remain disabled unless they are specifically needed for debugging purposes.

A note about Open Connections and Open Pins

When displaying an Open Connection for a placed instance, if the specific source and/or sink pins are not yet known (or not yet specified by the router), the connection is rendered to/from the center of the placed instance instead of to /from a specific pin. Likewise, when specific pins are not known, the **Open Pins** squares (red by default) are rendered in the center of the placed instance instead of on a specific pin. (Open Connections and Open Pins are not, of course, rendered for unplaced instances.)

Be aware that in a placed design that has not yet been routed, all nets are considered Open Connections.

Enabling **Open Pins** can make it much easier to find unrouted portions of a mostly routed design when zoomed out. But be aware this might be overwhelming on a large design that has been placed but not yet routed. (Every unrouted net is considered Open, thus in an unrouted design, every endpoint of every net displays an Open Pin square, merging into a single large mass of color when zoomed out.)

Objects in the ACE Selection Set are always visible

By default, any/all objects in the current ACE selection set (as shown in the Selection view (see page 136)) are always visible in the Floorplanner, regardless of the chosen "Layers" filter settings. This means even if the **Instances** layer is disabled, any Instances in the current ACE selection set are still painted in the Selected Instances color (by default, a bright green). Details of this behavior may be configured on the Floorplanner View Colors and Layers Preference Page (see page 190).

In addition to the layers listed in the table, there are several other types of information displayed in the Floorplanner — enabling and disabling the display of these other types of information is controlled from other views. For example: the visibility of individual Clock Regions is controlled from the Clock Regions view (see page 36); the visibility of individual Placement Regions is controlled from the Placement Regions view (see page 120); and the visibility of individual Critical Paths is controlled from the Critical Paths view (see page 48).

Highlighting

Special colored Highlighting of objects in the Floorplanner is possible via Tcl (see the `highlight` command) and/or may be triggered via associated highlighting actions in most of the other Views in the Floorplanner Perspective. Highlighted objects are only visible in the Floorplanner if the appropriate Layer is enabled, and the highlight color is only used if the object is not currently a member of the ACE Selection set. (By default, the Selection color takes precedence over the highlight color, which in turn takes precedence over the default color of the object. Further information about precedence of these states for Instances can be found under Instance States (see page 243), and can be partially reconfigured in the Floorplanner View Colors and Layers Preference Page (see page 190). Additional info regarding highlighting can be found at Highlighting Objects in the Floorplanner View (see page 315).)

Selection

The ( ) Selection Options control the selection of objects with the mouse in the Floorplanner view. Selected objects are added to the ACE Selection Set, and displayed appropriately in the Selection View (see page 136).

Table 24: *Selection Options*

Option	Default	Description
Instances	Enabled	Enables visible instances to be selected. If not checked, instances in the selection region are not added to the ACE selection set.
Nets	Enabled	Enables visible nets to be selected. If not checked, nets in the selection region are not added to the ACE selection set.

Option	Default	Description
Pins	Disabled	Enables visible user design pins to be selected. If not checked, pins in the selection region are not added to the ACE selection set.
Paths	Disabled	Enables visible paths to be selected. If not checked, paths in the selection region are not added to the ACE selection set.
Sites	Disabled	Enables visible sites to be selected. If not checked, sites in the selection region are not added to the ACE selection set.
Action		
Select	Enabled	Controls the action applied to objects in the selection region. Causes the objects to be added to the current ACE selection set.
Deselect	Disabled	Controls the action applied to objects in the selection region. Causes the objects to be removed from the current ACE selection set.
Remove Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes the placed instances to be un-placed.
Fix Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes the soft-placed instances to attempt to have fixed placement at the same site.
Un-fix Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes any fixed-placed instances to change to soft placement at the same site.

Selection actions with the mouse are filtered by Layers visibility

If **Instances** is checked under **Selection** but not under **Layers**, it is not possible to perform selection actions upon instances in the Floorplanner View using the mouse.

For example, this allows performing selection actions on only clock routes or only non-clock routes as desired, by simply setting the **Layers** filters appropriately.

Placement

The ( ) Placement Options control the drag-and-drop placement behavior in the Floorplanner view.

Table 25: *Placement Options*

Option	Default	Description
Group Placement	Disabled	[Expert Functionality] This option controls whether single instances or groups of instances are placed with the drag-and-drop action of the Placement Tool. Group Placement requires a group of instances to be in the ACE Selection Set prior to initiating drag and drop. Group Placement only succeeds in very specific circumstances, thus this setting should only be enabled by expert users who understand the caveats.
Fixed Placement	Enabled	This option controls whether the drag-and-drop placement of an instance should be considered fixed or soft. Fixed placements are not changed by the placer. Soft placements are taken as a placement hint and might be changed by the placer.

Caution

When pre-placing objects (for a pre-placement constraints `.pdc` file), **Fixed Placement** should always be enabled.

Tool Tip Text

The ( ) Tooltip options control the tooltip content while hovering over visible objects in the Floorplanner view.

Table 26: *Tooltip Options*

Option	Default	Description
Allow Tooltips	Enabled	Allows enabling/disabling Tooltip support for the Floorplanner without needing to toggle all the individual checkboxes.
Instance Names	Enabled	Includes the names of all placed instances under the current mouse position in the tooltip text.
Port Names	Enabled	Includes the RTL port names of placed instances under the current mouse position in the tooltip text.
Net Names	Enabled	Includes all net names under the current mouse position in the tooltip text.
Pin Names	Disabled	Includes all user design pin names under the current mouse position in the tooltip text.
Site Names	Enabled	Includes all leaf site names under the current mouse position in the tooltip text.
Site Types	Enabled	Includes the site cell type of each leaf site under the current mouse position in the tooltip text.

Option	Default	Description
Site Pin Names	Disabled	Includes all site pin names under the current mouse position in the tooltip text.
Device Port Names	Enabled	Includes the top-level port names of the target device under the current mouse position in the tooltip text.
Subtile Coordinates [1]	Enabled	Includes the subtile coordinates under the current mouse position in the tooltip text.
Partition Names	Enabled	Includes all partition names under the current mouse position in the tooltip text.
Clusters	Enabled	Includes the name of the cluster under the current mouse position in the tooltip text.

Table Notes

1. Subtile coordinates may be used with placement region commands on the Tcl command line.

Tooltips are filtered by Layers Visibility

If **Instance Names** is checked under "Tool Tip Text" but **Instances** is not checked under "Layers", it is not possible to see instance names in the tooltips.

Label

The ( ) Label options control the text labels on objects in the Floorplanner view.

Table 27: *Label Options*

Option	Default	Description
None	Enabled	This option disables label display.
Instance Names	Disabled	This option displays the instance names on placed instances.
Port Names	Disabled	This option displays the RTL port names on placed instances.
Site Names	Disabled	This option displays the full site names on each leaf site.
Site Types	Disabled	This option displays the site cell type on each leaf site.
Device Port Names	Disabled	This option displays the top-level port names of the target device connected to the I/O site.

Flow View

The Flow view provides a hierarchical view of Flow Steps (see page 221) that can be performed on the Active Project and Implementation (see page 221). From here, flow steps can be run and Flow Status (see page 225) viewed. Flow steps are not able to run unless an active implementation is selected in the Projects View (see page 125). When running flow steps, the implementation options (see page 103) of the active implementation are used to govern the behavior of the flow. Be aware that altering the value of an implementation option will clear the flow state of all downstream flow steps, changing them from the **Complete** state back to **Incomplete**.

By default, the Flow view is included in the Projects perspective (see page 24). To add it to the current perspective, click **Window** → **Show View** → **Other...** → **Achronix** → **Flow**.

For more details, see the Flow (see page 221) concept and the tasks for Running the Flow (see page 280).

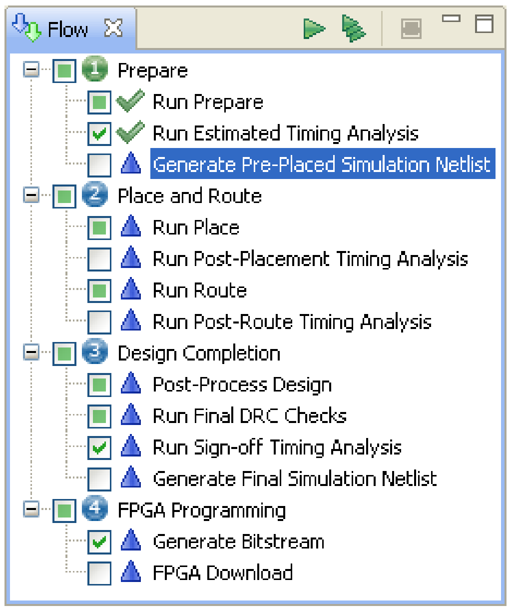


Figure 13: *Flow View Example*

Table 28: *Flow View Icons*

State	Flow Category	Flow Step
Incomplete		
Running		
Complete		
Disabled		
Warning		
Error		

Note

If the ( ) icon appears on a Flow Category or Flow Step, this typically means ACE has detected changes to project source files, where the current source files on disk no longer match the design currently in memory. See Detecting Changes to Project Source Files (see page 297).

Table 29: *Flow View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	Run Flow	Y		Runs all the enabled flow steps sequentially from the beginning of the flow.
	Resume Flow	Y		Resumes running the flow from the last completed flow step. If no flow steps have been attempted yet, this action behaves identically to **Run Flow**.
	Show Multiprocess View	Y		Launches the Multiprocess View (see page 83), which allows managing multiple runs of the flow in parallel.
	Stop flow (1)	Y	Y	Stops the execution of any flow steps after the currently running flow step. Also attempts to interrupt the currently running flow step if possible.
	Run Selected Flow Step (2)		Y	Runs the selected flow step, also running any required preceding flow steps that have not yet run. Preceding flow steps that are enabled but not required are skipped.
	Re-Run Flow		Y	Runs all the enabled flow steps sequentially from the beginning of the flow. Behavior is now identical to **Run Flow**.
	Re-Run Flow with `-ic init`(3)		Y	Behaves identically to **Re-Run Flow**, unless Incremental Compilation is enabled. If Incremental Compilation is enabled, this additionally forces a full recompile of all partitions; any prior partition state is ignored (and overwritten).
	Clear Flow (4)		Y	Issues a `clear_flow` (see page 538)TCL command. All flow categories and flow steps with the state of Complete or Error are reset to the state of Incomplete. Additionally, the state of the current active project and implementation are cleared, as if they had not yet been run through the flow.
	Create Flow Step (5)		Y	Displays an interactive dialog for creating a user-defined flow step (the dialog includes a prompt for which a single Tcl command should be invoked for this step) at the selected location within the flow.
	Remove Flow Step		Y	Removes a selected user-defined flow step. Only steps the user has created with **Create Flow Step** may be removed; this action cannot be used to remove "reserved" steps.

Table Notes

1. Some flow steps, such as FPGA Download, are currently unable to be interrupted while running.
2. Double-clicking a flow step is equivalent to this action.
3. **Caution:** This action is only relevant when Using Incremental Compilation (Partitions) (see page 363). If Incremental Compilation is disabled, this action behaves identically to **Re-Run Flow**.
4. Clear Flow does not remove any prior saved state from the hard drive. Any prior saved state may subsequently be (re-)loaded, including any partition state for incremental compilation.
5. **Caution:** Creation of user-defined flow steps is only recommended for advanced users.

> **Warning!**
>
> **The current Flow Mode setting impacts which Flow Steps are executed.**
> The implementation option for Flow Mode (see page 226) affects which flow steps are executed during the **Run Flow**, **Resume Flow**, **Re-Run Flow**, and **Re-Run Flow with `"-ic init"`** actions (or related Tcl commands).

HW Demo View

> **Warning!**
>
> - **The JTAG connection must be configured before using the HW Demos.**
> ACE interacts with the FPGA using the JTAG interface through a Bitporter pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the HW Demo functionality. The configuration is managed using the Configure JTAG Connection Preference Page (see page 186). See Configuring the JTAG Connection (see page 331) for more details.
> - **The DCC connection must be configured before using the HW Demos.**
> ACE interacts with the HW Demo designs (and reference designs) using the DCC interface to the FPGA through a USB cable (not the Bitporter). This interface must be properly configured in ACE before using the HW Demo functionality. The configuration is managed using the Configure DCC Connection Preference Page (see page 185). See Configuring the DCC Connection (see page 330) for more details.

The HW Demo view provides a graphical interface for demonstrating particular aspects of a user selected device, using provided sample designs. These sample designs are typically provided as self-documenting overlays for the standard ACE installation.

By default, the HW Demo view is included in the HW Demo Perspective (see page 24). To access the HW Demo view from any other perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **HW Demo**.

Before any demo overlays are installed, there are no demo designs available. In that case, the view will display minimal information:

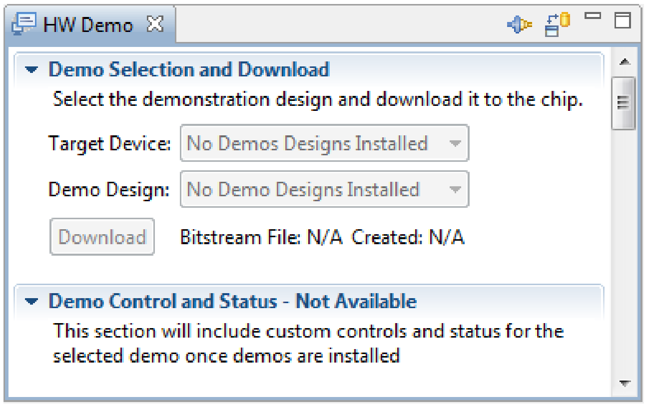


Figure 14: ***HW Demo View with No Demo Designs Example***

After demos are installed as ACE overlays, from this view the status of various board components can be observed updating in real-time as the demonstration design is running on the connected board. For example, in a basic fabric demonstration, when an associated DIP switch is changed, it is reflected in the view display; likewise when an LED on the board changes state (on/off) it is reflected in the view. Individual memory locations may be read and examined, and new values may be entered and "pushed" out to the target device.

Most hardware demos (including reference designs) are designed to show the features of the various hard IP blocks integrated into the target Achronix FPGA. Each demo or reference design installation package comes with associated documentation specific to the design.

Talk to your Achronix Marketing contact or FAE to request access to the demos appropriate to your development board.

See also: Running the HW Demo (see page 361).

Table 30: *HW Demo View Toolbar Buttons*

Icon	Action	Description
	Configure DCC Interface	Opens the preferences dialog to the Configure DCC Connection Preference Page (see page 185).
	Configure JTAG Interface	Opens the preferences dialog to the Configure JTAG Connection Preference Page (see page 186).

Table 31: *HW Demo View Options*

Option	Description
Demo Selection and Download	
Target Device	List of FPGA devices that have demonstration designs.
Demo Design	List of demonstration designs for the currently selected device.
Download	Loads the currently selected demonstration design into the attached board.
Board Status	
LED State	Visually represents relevant LEDs from the attached board. When an LED changes state on the board, it is reflected in the view LED display. Clicking an individual LED in the view causes the corresponding LED on the attached board to toggle its state.
DIP Switch State	Visually represents the eight DIP switches from the attached board. When a switch changes state on the board, it is reflected in the DIP switch display. Clicking an individual switch in the view does *not* cause the corresponding switch on the attached board to toggle its state.
Device State	Displays DCC connection status and demo version number.
Demo Control and Status	
Each specific demonstration design has a simple user interface that is presented in the bottom section of the view. An example interface might provide a facility for reading and writing values to user specified addresses.	

> **Note**
>
> The Board Status section might not be present in all HW Demo (and reference) designs.

I/O Designer Toolkit Views

The I/O Designer Toolkit views provide a set of fully integrated I/O ring design tools. With these tools it is possible to:

- Combine I/O ring IP configuration (`.acxip`) files into a complete I/O ring design
- View and update dynamic I/O ring resource utilization
- View and update (using drag and drop) dynamic I/O ring floorplan layout
- View dynamic I/O ring package ball layout and pin assignment report
- Automatically complete the following:
  - Full I/O ring final DRC, in real time
  - Full I/O ring timing closure, in real time
  - Full I/O ring place and route, in real time
- Generate complete package ball pin assignment, power, and utilization reports
- Generate a full I/O ring simulation model that is 100% correctly configured, and has wrappers tailored specifically to the user design
- Generate pin placements PDC, Verilog wrappers, and port lists for the core user design
- Generate the full I/O ring bitstream
- Quickly and easily combine existing I/O ring IP configuration (`.acxip`) files from an existing ACE project into new ACE projects to create multiple designs

> **Caution!**
>
> The I/O Designer views and features are only applicable to specific Achronix Speedster devices, such as the Speedster7t AC7t1500.

I/O Designer Toolkit Views

The views in the I/O Designer Toolkit are:

- I/O Utilization View (see page 67)
- I/O Package Diagram View (see page 68)
- I/O Pin Assignment View (see page 69)
- I/O Core Pin Assignment View (see page 70)
- I/O Layout Diagram View (see page 72)

I/O Ring Design File Generation

Clicking the ( ) **Generate I/O Ring Design Files** toolbar button opens the Generate I/O Ring Design Files Dialog (see page 176), which allows selection of the output directory for all the customized I/O ring design files, including:

- Complete package ball pin assignment, power, and utilization reports
- Pin placements PDC, Verilog wrappers, and port lists for the core user design

- The full I/O ring bitstream, which is automatically combined with the core user design bitstream in ACE at the end of the normal place-and-route flow for the core user design
- Customized I/O ring simulation files, including Verilog wrappers for the top-level and I/O ring configuration data

> **Batch Mode Support**
>
> I/O ring design files may also be generated in batch mode for a given ACE project by calling the generate_ioring_design_files (see page 554) Tcl command. This command loads up all the I/O ring IP configuration (`.acxip`) files from an existing ACE project, and performs full design rule checks prior to generating the output files. I/O ring IP configuration files can also be edited in a text editor to support batch mode configuration prior to calling the `generate_ioring_design_files` Tcl command.

Table 32: *I/O Designer Toolbar Buttons*

Icon	Description
	Opens the Generate I/O Ring Design Files Dialog (see page 176), which allows selecting the directory into which the customized I/O ring design files are generated.

See also Creating an IP Configuration (see page 306) and Adding Source Files (see page 272).

I/O Utilization View

The I/O Utilization view provides a combined utilization summary of the active ACE project I/O ring IP configuration (`.acxip`) files, including shared resources such as clocks. Each resource type is summarized in a table inside an expandable section. These tables can be used for navigating between various IP configuration files in the project. Configuration errors are also summarized in the Status column for each row. Right-clicking a row brings up a context menu of actions that can be performed on the IP in that row. Double-clicking the table row opens the source IP configuration file for the data in that row.

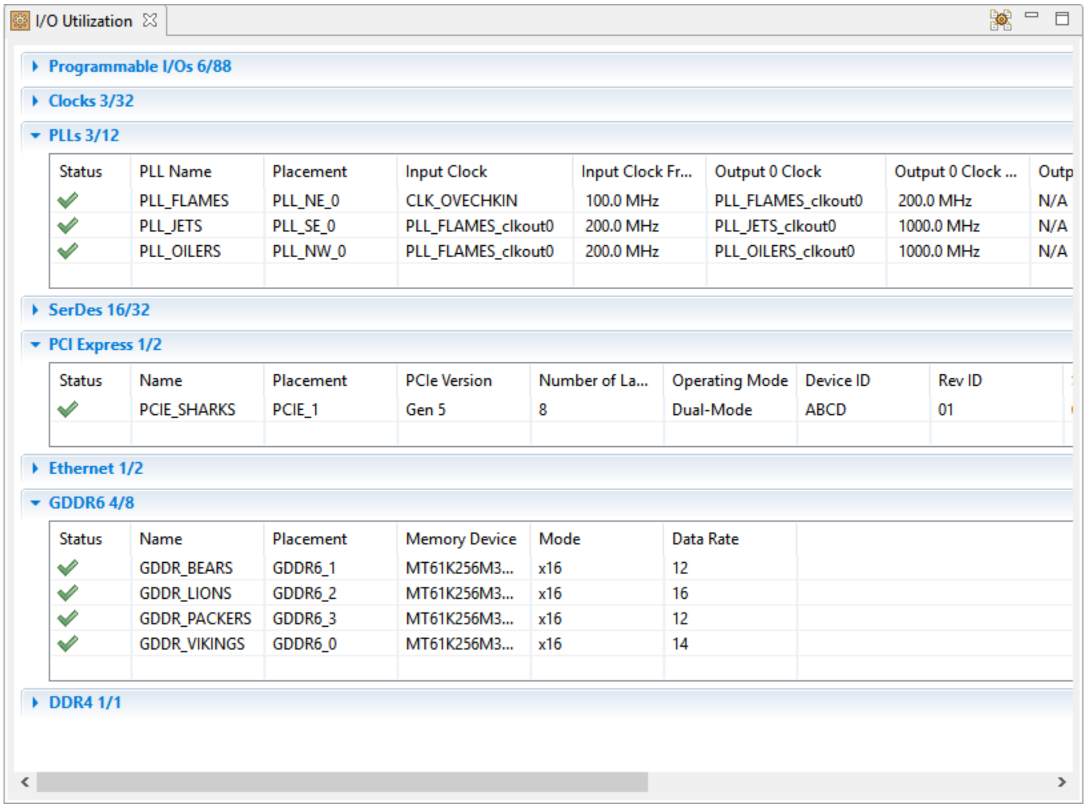


Figure 15: ***I/O Designer View (Utilization Tab)***

Table 33: ***I/O Designer View Actions***

Icon	Action	Description
	Open IP	Opens the selected IP file in an editor within ACE.
	Clone IP	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP	Renames the selected IP.
	Remove IP from project	Allows removal of the selected IP project. See also remove_project_ip (see page 575).
	Show in file manager	Opens the operating system default file manager to the directory containing the IP file.

I/O Package Diagram View

The I/O Package Diagram view shows a live diagram of the target package balls and all I/O ring user design top-level pin ball assignments. Click and drag to pan the diagram and use the mouse wheel to zoom in and out. Package balls with yellow fill indicate placed user design pins on those package balls. Tooltip text provides extra information about each package ball location.

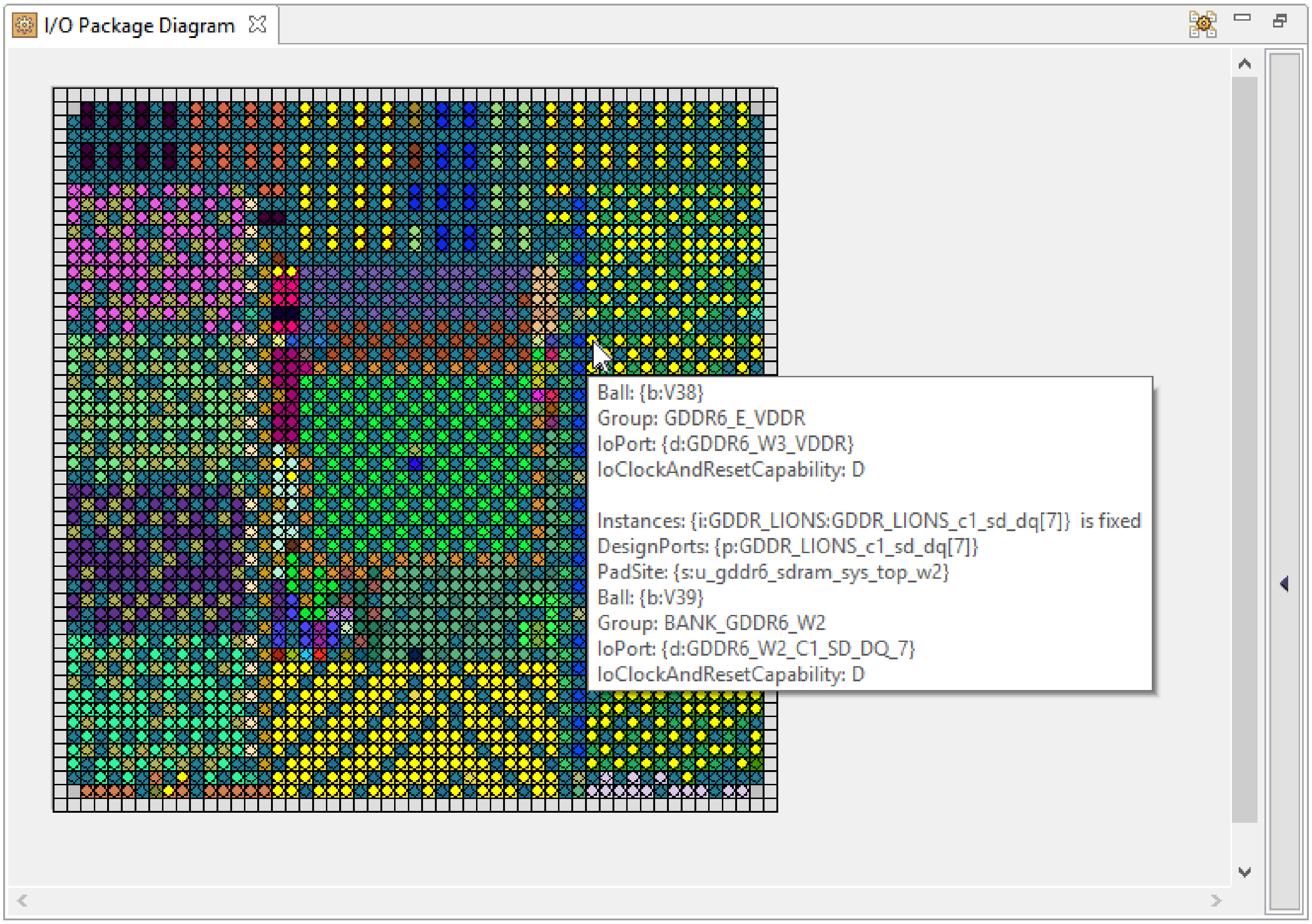
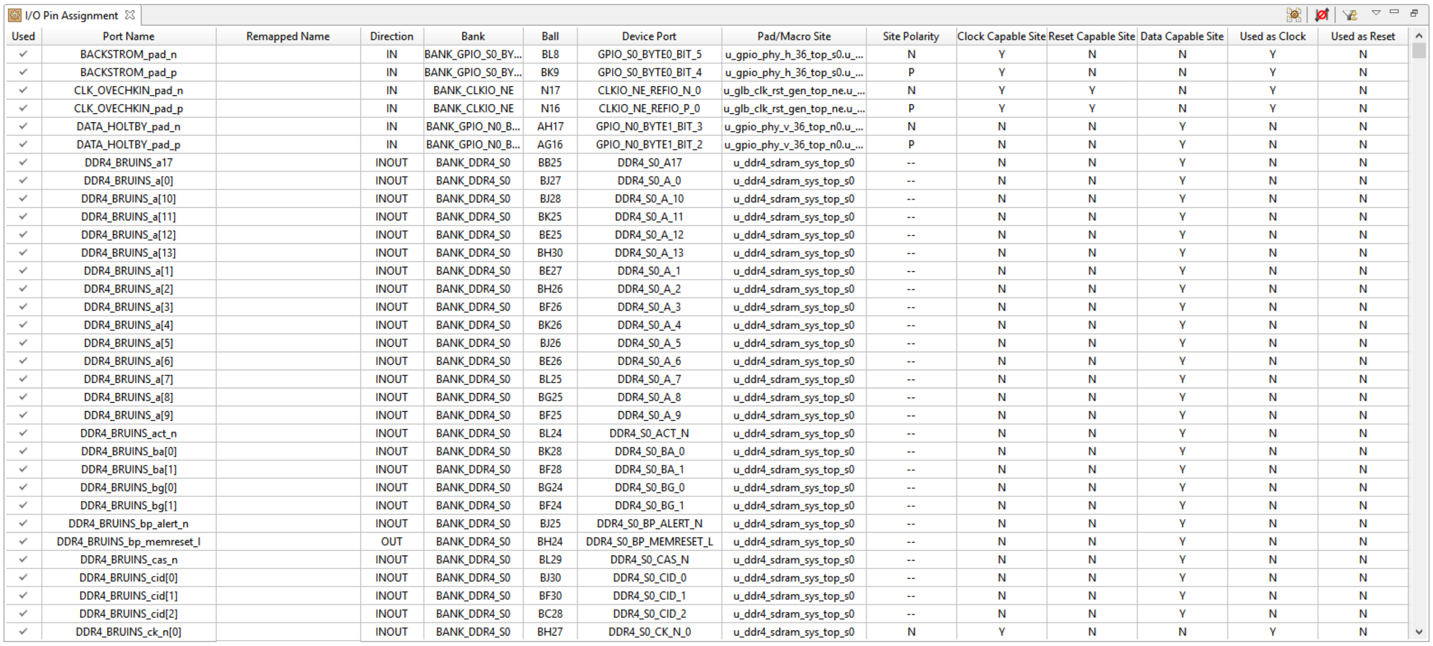


Figure 16: ***I/O Package Diagram View***

I/O Pin Assignment View

The I/O Pin Assignment view shows a live table of I/O ring user design top-level pin assignment information, including user design port name, I/O bank, package ball, top-level device port name, and pad/macro site name (for debugging in the full-chip simulation hierarchy). Columns can be sorted by left-clicking on the column headers. Columns can be filtered using the **Toggle Filter Row Visibility** button.



I/O Pin Assignment

Used	Port Name	Remapped Name	Direction	Bank	Ball	Device Port	Pad/Macro Site	Site Polarity	Clock Capable Site	Reset Capable Site	Data Capable Site	Used as Clock	Used as Reset
✓	BACKSTROM_pad_n		IN	BANK_GPIO_S0_BY...	BL8	GPIO_S0_BYTE0_BIT_5	u_gpio_phy_h_36_top_s0.u_...	N	Y	N	N	Y	N
✓	BACKSTROM_pad_p		IN	BANK_GPIO_S0_BY...	BK9	GPIO_S0_BYTE0_BIT_4	u_gpio_phy_h_36_top_s0.u_...	P	Y	N	N	Y	N
✓	CLK_OVECHKIN_pad_n		IN	BANK_CLKIO_NE	N17	CLKIO_NE_REFIO_N_0	u_glb_clk_rst_gen_top_ne.u_...	N	Y	Y	N	Y	N
✓	CLK_OVECHKIN_pad_p		IN	BANK_CLKIO_NE	N16	CLKIO_NE_REFIO_P_0	u_glb_clk_rst_gen_top_ne.u_...	P	Y	Y	N	Y	N
✓	DATA_HOLTBY_pad_n		IN	BANK_GPIO_N0_B...	AH17	GPIO_N0_BYTE1_BIT_3	u_gpio_phy_v_36_top_n0.u_...	N	N	N	Y	N	N
✓	DATA_HOLTBY_pad_p		IN	BANK_GPIO_N0_B...	AG16	GPIO_N0_BYTE1_BIT_2	u_gpio_phy_v_36_top_n0.u_...	P	N	N	Y	N	N
✓	DDR4_BRUINS_a17		INOUT	BANK_DDR4_S0	BB25	DDR4_S0_A17	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[0]		INOUT	BANK_DDR4_S0	BJ27	DDR4_S0_A_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[10]		INOUT	BANK_DDR4_S0	BJ28	DDR4_S0_A_10	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[11]		INOUT	BANK_DDR4_S0	BK25	DDR4_S0_A_11	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[12]		INOUT	BANK_DDR4_S0	BE25	DDR4_S0_A_12	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[13]		INOUT	BANK_DDR4_S0	BH30	DDR4_S0_A_13	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[1]		INOUT	BANK_DDR4_S0	BE27	DDR4_S0_A_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[2]		INOUT	BANK_DDR4_S0	BH26	DDR4_S0_A_2	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[3]		INOUT	BANK_DDR4_S0	BF26	DDR4_S0_A_3	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[4]		INOUT	BANK_DDR4_S0	BK26	DDR4_S0_A_4	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[5]		INOUT	BANK_DDR4_S0	BJ26	DDR4_S0_A_5	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[6]		INOUT	BANK_DDR4_S0	BE26	DDR4_S0_A_6	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[7]		INOUT	BANK_DDR4_S0	BL25	DDR4_S0_A_7	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[8]		INOUT	BANK_DDR4_S0	BG25	DDR4_S0_A_8	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[9]		INOUT	BANK_DDR4_S0	BF25	DDR4_S0_A_9	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_act_n		INOUT	BANK_DDR4_S0	BL24	DDR4_S0_ACT_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ba[0]		INOUT	BANK_DDR4_S0	BK28	DDR4_S0_BA_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ba[1]		INOUT	BANK_DDR4_S0	BF28	DDR4_S0_BA_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bg[0]		INOUT	BANK_DDR4_S0	BG24	DDR4_S0_BG_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bg[1]		INOUT	BANK_DDR4_S0	BF24	DDR4_S0_BG_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bp_alert_n		INOUT	BANK_DDR4_S0	BJ25	DDR4_S0_BP_ALERT_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bp_memreset_l		OUT	BANK_DDR4_S0	BH24	DDR4_S0_BP_MEMRESET_L	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cas_n		INOUT	BANK_DDR4_S0	BL29	DDR4_S0_CAS_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[0]		INOUT	BANK_DDR4_S0	BJ30	DDR4_S0_CID_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[1]		INOUT	BANK_DDR4_S0	BF30	DDR4_S0_CID_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[2]		INOUT	BANK_DDR4_S0	BC28	DDR4_S0_CID_2	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ck_n[0]		INOUT	BANK_DDR4_S0	BH27	DDR4_S0_CK_N_0	u_ddr4_sdram_sys_top_s0	N	Y	N	N	Y	N

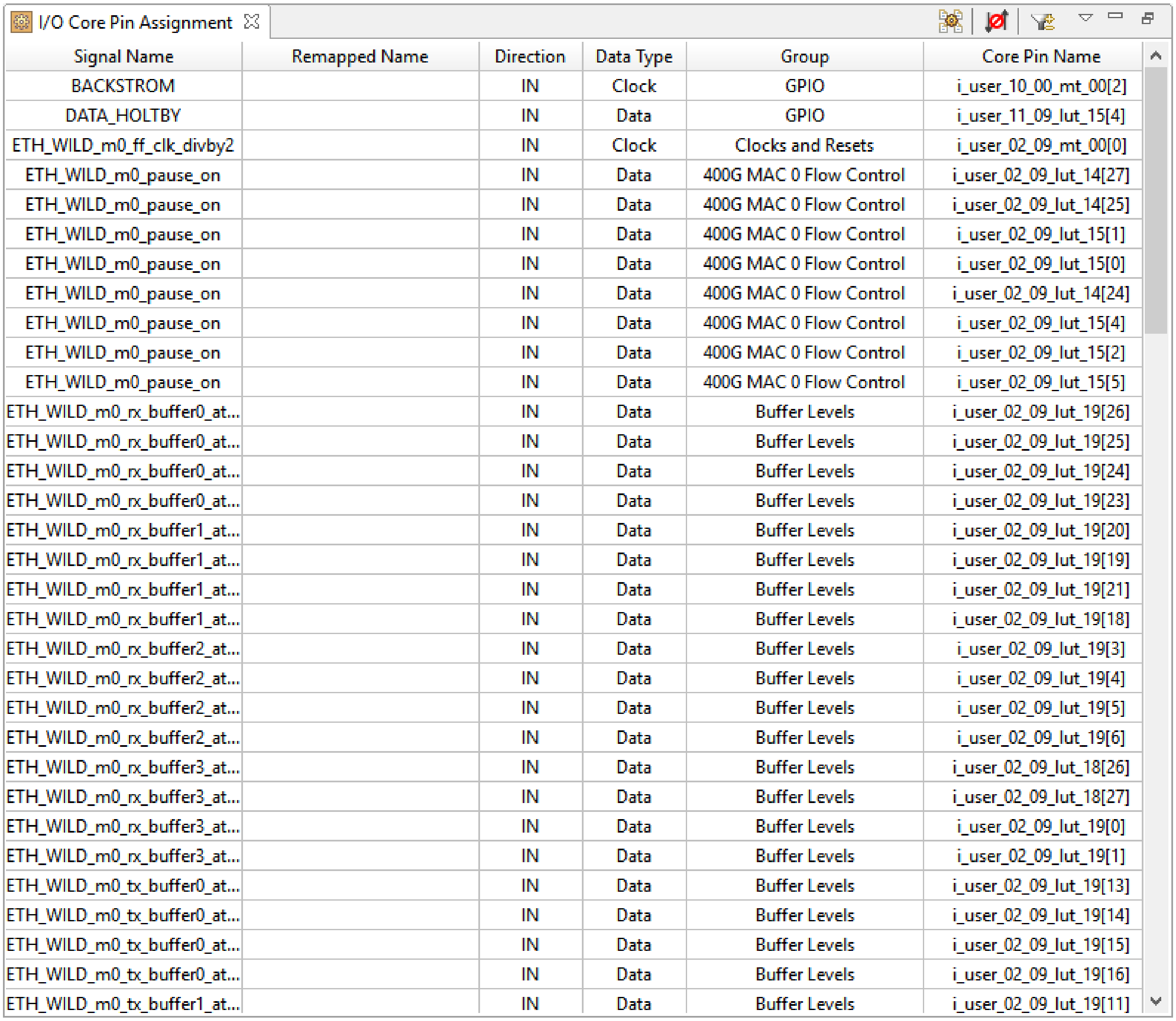
Figure 17: ***I/O Pin Assignment View***

Table 34: ***I/O Pin Assignment View Buttons***

Icon	Description
	Generate I/O Ring Design Files.
	Clear Sorting.
	Toggle Filter Row Visibility.
	Remap Port/Signal Name (available in **Remapped Name** column right-click menu).

I/O Core Pin Assignment View

The I/O Core Pin Assignment view shows a live table of I/O ring user design top-level core pin assignment information, including user design signal name, direction, data type, group, and core pin name. Columns can be sorted by left-clicking the column headers. Columns can be filtered using the **Toggle Filter Row Visibility** button.



I/O Core Pin Assignment

Signal Name	Remapped Name	Direction	Data Type	Group	Core Pin Name
BACKSTROM		IN	Clock	GPIO	i_user_10_00_mt_00[2]
DATA_HOLTBY		IN	Data	GPIO	i_user_11_09_lut_15[4]
ETH_WILD_m0_ff_clk_divby2		IN	Clock	Clocks and Resets	i_user_02_09_mt_00[0]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[27]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[25]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[1]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[0]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[24]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[4]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[2]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[5]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[26]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[25]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[24]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[23]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[20]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[19]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[21]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[18]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[3]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[4]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[5]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[6]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_18[26]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_18[27]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[0]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[1]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[13]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[14]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[15]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[16]
ETH_WILD_m0_tx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[11]

Figure 18: ***I/O Core Pin Assignment View***

Table 35: ***I/O Core Pin Assignment View Buttons***

Icon	Description
	Generate I/O Ring Design Files.
	Clear Sorting.
	Toggle Filter Row Visibility.

Icon	Description
	Remap Port/Signal Name (available in **Remapped Name** column right-click menu).

I/O Layout Diagram View

The I/O Layout Diagram view shows an interactive floorplan of the target device. Empty IP Sites are shown in white. Sites with IP legally placed on them are shown in green. Sites with IP placement overlap violations are shown in red. IP may be dragged and dropped to adjust placement within the diagram. The holding down the CTRL key while dragging creates a clone of an IP at another site. Double-clicking a placed IP opens an editor for that IP.

Changes to placement in the diagram result in updates to the source IP configuration (`.acxip`) files. Tooltip text provides extra information about each IP site.

Right-clicking a site brings up a context menu of actions that can be performed on that site, or the IP placed on that site.

Figure 19: ***I/O Designer View (Layout Tab)***

Table 36: ***I/O Designer View Actions***

Icon	Action	Description
	Open IP	Opens the selected IP file in an editor within ACE.
	Create new IP here	Creates a new IP at the chosen site.

Icon	Action	Description
	Clone IP	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP	Renames the selected IP.
	Add IP to another project...	Adds the selected IP to another project in the ACE workspace.
	Add copies of IP to another project...	Adds a copy of the selected IP to another project in the ACE workspace.
	Remove IP from project	Allows removal of the selected IP project. See also remove_project_ip (see page 575)

By default, if one or more IP editors are currently open, the diagram is configured to display a yellow highlight indicating the currently active IP editor.

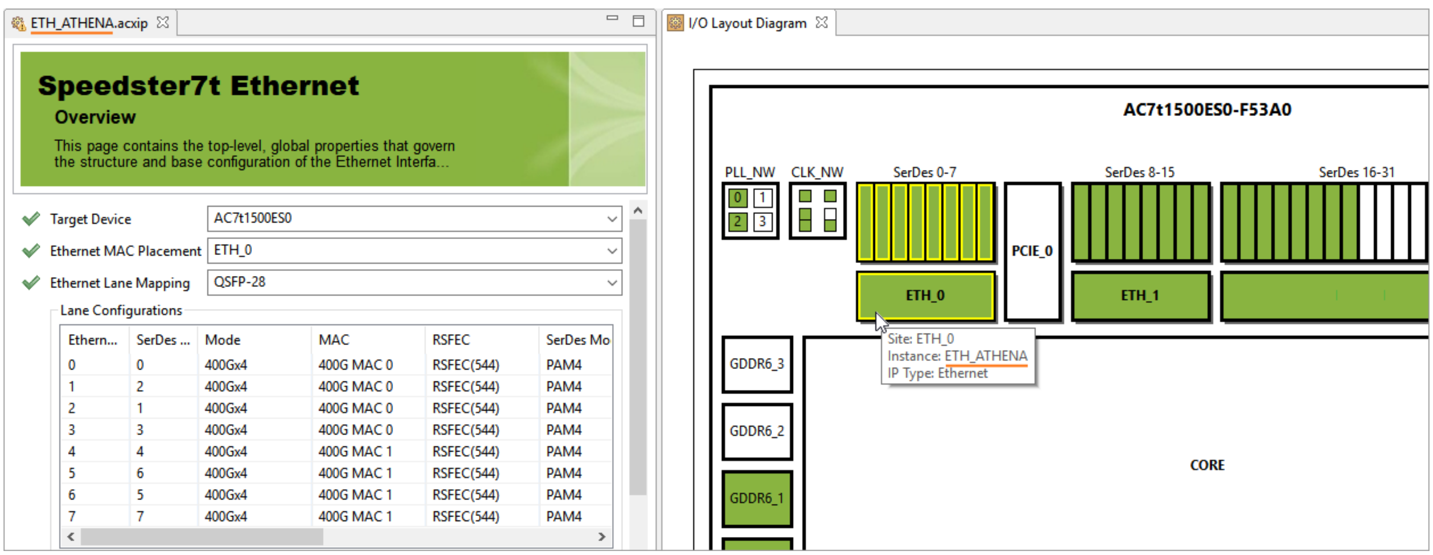


Figure 20: *I/O Layout Diagram View Currently Active Highlight Example*

The **I/O Designer** page in the Preferences can be used to adjust the highlight color, or to hide it.

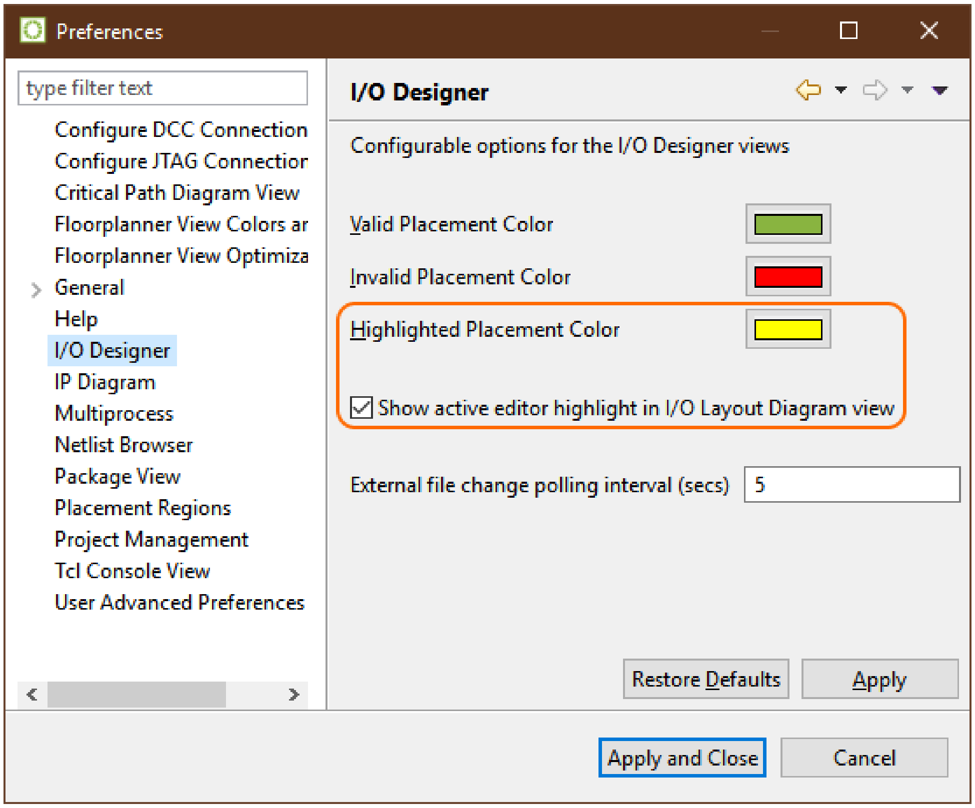


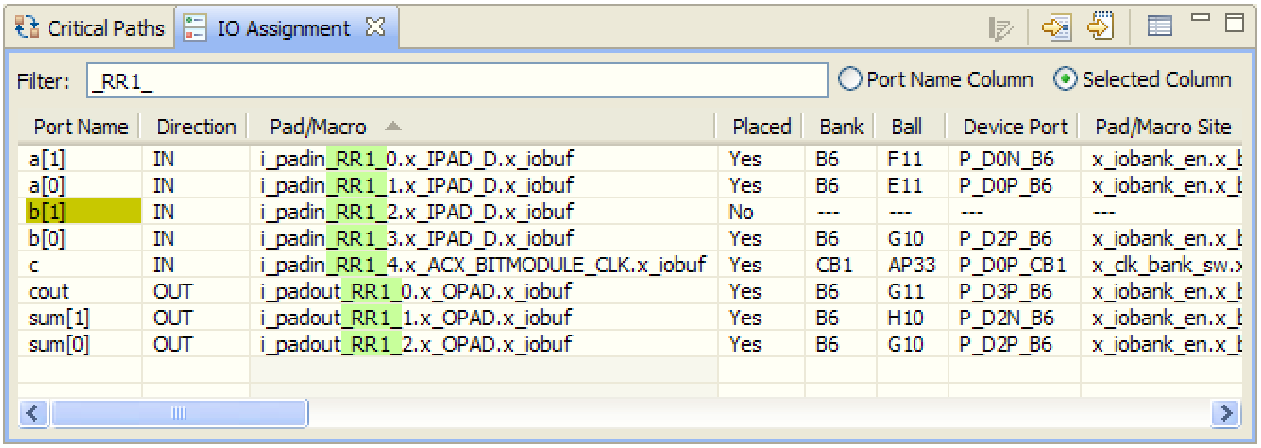
Figure 21: ***I/O Designer Highlighted Placement Color Preference Example***

IO Assignment View

> **The IO Assignment View is only applicable to Speedster FPGAs**
>
> This View should be ignored when developing for other Achronix product types.

The IO Assignment view provides a tabular representation of the properties of the I/O instances used in the current design. The view remains empty until the currently active Implementation has been prepared (completed the **Run Prepare** flow step).

By default, the IO Assignment view is included in the Floorplanner Perspective (see page 25). To add the view to the current perspective, select **Window** → **Show View** → **Other...** → **IO Assignment**. See also: Managing I/Os (see page 431).


Critical Paths | IO Assignment

Filter: _RR1_ ◯ Port Name Column ◉ Selected Column

Port Name	Direction	Pad/Macro	Placed	Bank	Ball	Device Port	Pad/Macro Site
a[1]	IN	i_padin_RR1_0.x_IPAD_D.x_iobuf	Yes	B6	F11	P_D0N_B6	x_iobank_en.x_l
a[0]	IN	i_padin_RR1_1.x_IPAD_D.x_iobuf	Yes	B6	E11	P_D0P_B6	x_iobank_en.x_l
b[1]	IN	i_padin_RR1_2.x_IPAD_D.x_iobuf	No	---	---	---	---
b[0]	IN	i_padin_RR1_3.x_IPAD_D.x_iobuf	Yes	B6	G10	P_D2P_B6	x_iobank_en.x_l
c	IN	i_padin_RR1_4.x_ACX_BITMODULE_CLK.x_iobuf	Yes	CB1	AP33	P_D0P_CB1	x_clk_bank_sw.x
cout	OUT	i_padout_RR1_0.x_OPAD.x_iobuf	Yes	B6	G11	P_D3P_B6	x_iobank_en.x_l
sum[1]	OUT	i_padout_RR1_1.x_OPAD.x_iobuf	Yes	B6	H10	P_D2N_B6	x_iobank_en.x_l
sum[0]	OUT	i_padout_RR1_2.x_OPAD.x_iobuf	Yes	B6	G10	P_D2P_B6	x_iobank_en.x_l

Figure 22: ***IO Assignment View Example***

Note

In the figure above, a filter of "_RR1_" is applied to the Selected Column. The Selected Column is **Pad/Macro**, subtly identified by the grey highlight in the unpopulated rows and the sort arrow in the column header. The content of the table is being limited to those rows that match the filter, and all the remaining filtered rows have highlighted (in green) the portion of the column data that matches the filter. The port "b[1]" also has its **Port Name** cell highlighted in dark yellow to indicate that there is a warning for the port. The warning details are available via a tooltip on that cell (not shown). In this case, the warning is that the port has not been placed yet, as can be seen by the value of "No" in the **Placed** column.

The following are ways to alter the presentation of the data in the IO Assignment table:

- **Column Resizing**
  To change the width of a column:
  1. Place the mouse cursor over the boundary between columns.
  2. At this point the mouse cursor should change to indicate resizing is possible.
  3. Simply left-click and drag left or right to resize the column to the desired width.
  4. Release the mouse button.
- **Column Reordering**
  To change the order of the columns in the table:
  Left-click and hold any column name
  1. Drag left or right to move the column between any other pair of columns
  2. Release the left mouse button to insert the column header at the new location
  3. While dragging, the dragged column header is visible alongside the mouse cursor with a platform-specific visual indicator (a thick column header separator on Windows, a set of arrows pointing at the potential insertion location on Linux) showing where the column insertion is to occur if the mouse is released at the present cursor location.
- **Sorting by Column**
  To sort data rows by the contents of any column:
  1. Simply clicking the column header causes the rows tol be sorted by the data in that column
  2. Click the same column header a second time to reverse the sort order (ascending vs. descending)

- **Data Filtering**
  By using the **Filter:** textfield, the rows in the table may be filtered by the contents of any column. When a filter is applied, rows are not displayed if their data for the selected column does not match the current filter value. All displayed rows, in the selected column, contain text that matches the applied filter. The portion of the text that matches the filter is highlighted (in green). By default, the column being filtered is the **Port Name** column.
  To select an alternate column for the filter:
  1. Select the **Selected Column** radio button
  2. Click the corresponding column header
  3. The selected column is indicated in a platform-specific manner:
     - Windows: the column header has a small arrow indicating the sort direction of the selected column and any visible empty rows have the selected column in a slightly different color than the non-selected columns.
     - Linux: the selected column background color is different from the background color of non-selected columns.

  To stop filtering the table data and view the data for all I/Os:
  1. Simply clear/empty the **Filter:** textfield.

> **Note**
>
> Sorting column data and "selecting" a column for the filter are the same action. This means that except when filtering by the **Port Name** column, it is not possible to sort the table by one column and filter by a different column.

The following are ways to edit I/O data using the IO Assignment table:

- **Drag-and-Drop Placement**
  To change the I/O placement via drag-and-drop:
  1. Drag the I/O instances from the table to sites in the Floorplanner view (see page 53) or balls in the Package View (see page 112).
  2. The dragged instance is assigned to the drop location site/ball.
  3. The I/O location is immediately updated in all views.

  > **Warning**
  >
  > Changing I/O placement requires re-running Place and Route.

- **Editing Individual Table Cell Values**
  To edit the value displayed in a table cell (only available in certain columns):
  1. Select the cell and click or press **Enter**.
  2. The cell becomes editable, presenting a combo box of choices, or an in-place editable text field for free-form text entry.
  3. Change the value as desired and press **Enter** to commit the change.
  4. The changed value is validated, and if acceptable, the table is updated to reflect the new value.

> **Note**
>
> When presented with a combo box, the choices listed are filtered to show only values that are compatible with the rest of the configuration of that individual I/O instance.

- **Editing Electrical Values for One or More I/Os**
  To change the configuration of multiple I/O instances at once (i.e., for a bus):
  1. Select all the rows to edit.
  2. Select the **Configure Selected IOs** button in the view toolbar.
  3. The Configure Selected IOs Dialog appears to allow editing the configuration of the entire group of I/O instances.

Table 37: ***IO Assignment View Toolbar Buttons***

Icon	Action	Description
n/a	Filter:	This text field allows the entry of a search filter. The displayed contents of the IO Assignment table are restricted to those rows with data matching the filter. Text matching the filter is highlighted in the chosen column. Clear/empty the text field to stop filtering and restore the complete dataset to the table.
n/a	Port Name Column	Select this to apply the filter to the Port Name Column.
n/a	Selected Column	Select this to apply the filter to the currently selected column. The selected column is whichever column is managing the sort order for the table data. To select a column, click the table column header.
	Configure Selected IOs	Opens the Configure Selected IOs Dialog. Allows changing the electrical configuration values of the selected I/O(s). This action is only enabled when there is at least one row of data (at least one I/O) selected in the table.
	Save Changed Properties	Opens the Save Changed Properties Dialog (see page 168). Allows saving an `.sdc` file containing all properties changed (for all instances, not just I/Os) since the last time the Run Prepare flow step was executed. For more information, see Managing I/Os (see page 431).
	Generate Pin Assignment Report	Opens the Generate a Pin Assignment Report Dialog (see page 163). Allows generating a Pin Assignment Report (see page 227) with the same column configuration as the current IO Assignment table.
	Configure Columns	Opens the Configure Table Columns Dialog (see page 155). Allows changing which columns are shown in the IO Assignment table, allows the width of each column to be edited, and allows saving and loading favorite configurations of this table.

Table 38: ***I/O Assignment Table Columns***

Column	Editable	Description
Port Name		The top-level port name in the user design referencing this I/O.

Column	Editable	Description
Direction		The direction of this I/O: **IN**, **OUT**, or **INOUT**.
Pad/Macro		The instance name of the I/O buffer (pad) or the instance name of the I/O macro (e.g., SerDes) in the user design.
Placed		Indicates whether this I/O is placed.
Bank		The name of the physical I/O bank (or group) in which this I/O is placed.
Ball (1)	Yes	The package ball name upon which this I/O is placed.
Device Port		The top-level port name on the device to which this I/O is connected. (Only applies if an I/O is placed).
Pad/Macro Site		The site name on the device within which this I/O pad or macro instance is placed.
Site Polarity		The LVDS polarity of the site this I/O is placed on. **P** = Positive and **N** = Negative.
Bank Use		The usage type of the bank this I/O is placed in. **C** = Clock Bank, **G** = General Purpose Bank, **B** = Byte Lane Bank.
Clock Capable		Indicates whether the site this I/O is placed on supports Clock function.
Reset Capable		Indicates whether the site this I/O is placed on supports Reset function.
Data Capable		Indicates whether the site this I/O is placed on supports Data signals.
I/O Standard	Yes	The I/O Standard value for this I/O. The I/O Standard controls the voltage level and other settings for this I/O. This can be changed in ACE to a different I/O Standard value, but only if a compatible I/O standard exists that would not require re-synthesizing the design.
Voltage Level		The VDD voltage level for this I/O, set via the I/O Standard selection.
VREF Level		The VREF voltage level for this I/O, set via the I/O Standard selection.
Pad Polarity		The LVDS polarity of this I/O instance in the user design. This only applies for differential I/O buffers and is set in synthesis.
Pad DQ Capability		The DQ capability of this I/O instance in the user design. This only applies for byte lane I/Os.

Column	Editable	Description
Site DQ Capability		The DQ capability of the site this I/O instance is placed on. This only applies for byte lane I/Os.
Site DQ Domains		The list of accessible DQ Domains that can be reached from the site this I/O instance is placed on. This only applies for byte lane I/Os.
Drive Strength	Yes	The drive strength of this I/O. This only applies for outputs and inouts and a limited set of I/O Standards.
Hysteresis	Yes	The hysteresis setting of this I/O. This only applies for inputs and inouts and a limited set of I/O Standards.
Location		The location property can be specified in the user RTL for pre-placement of this I/O. However, it is recommended that the `set_placement` constraint is used instead. This cannot be changed in ACE after synthesis.
Open Drain	Yes	The open drain setting of this I/O. This only applies for outputs and inouts.
Slew	Yes	The slew rate of this I/O. This only applies for outputs and inouts and a limited set of I/O Standards.
On Die Termination	Yes	Indicates whether on die termination is enabled for this I/O. This only applies for inputs and inouts.
Termination Value	Yes	The termination value of this I/O. This only applies for inputs and inouts and may have different ranges for different I/O Standards. On Die Termination must be turned on for this to apply.
Keep Mode	Yes	The keep mode setting of this I/O. This only applies for inputs and inouts.
Used as Clock		Indicates whether this I/O is used as a clock in the design.
Used as Reset		Indicates whether this I/O is used as a reset in the design.
Clock Domain		The clock domain for this I/O.
Target Frequency (MHz)	Yes	The target frequency for the clock domain for this I/O. This is set via the SDC timing constraints, or is the HW limit if unspecified.

Table Notes

1. **Caution:** Editing the Ball changes the I/O placement, and requires that place and route be re-run for the design. This is not an electrical-only change.

IP Diagram View

The IP Diagram view is meant to provide a graphical visualization of the configuration of the IP currently being edited. As different IP configurations are selected (by selecting their Editor), the IP Diagram view contents change to reflect the selected IP configuration.

Some IP supports multiple pages of diagrams (e.g., a logic block diagram page and a placement diagram page). In these cases, there are multiple labeled tabs at the bottom of the IP Diagram view to allow switching diagram pages.

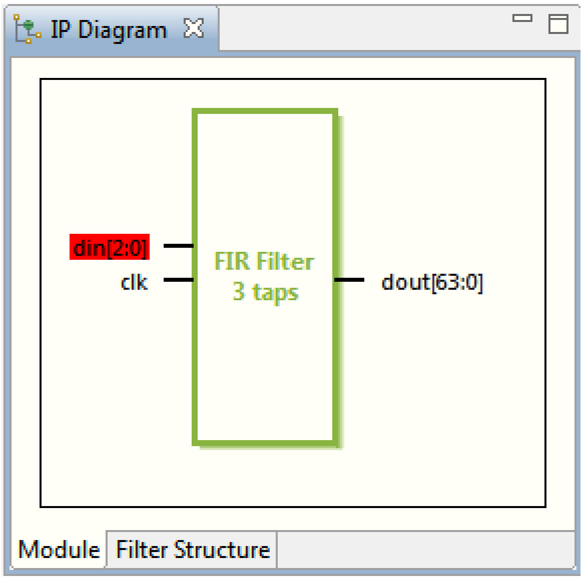


Figure 23: ***Example IP Diagram With din[2:0] Input Indicating a Configuration Error***

When a supported IP Configuration Editor is selected, the IP Diagram view shows a dynamic block diagram of the selected IP. Displayed labels change, and logic blocks may appear and disappear depending upon the configuration options currently selected in the IP Editor. Tool tips are available on all text displayed in the IP Diagram. Text representing Configuration Options with Warnings or Errors are displayed with appropriate colors to indicate the condition (by default, Warnings have a yellow background and Errors have a red background, though these colors may be overridden from the IP Diagram Preference Page (see page 199)).

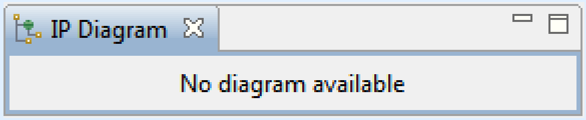
Clicking any text label in the IP Diagram immediately turns the IP Editor to the associated page so that the related Configuration Options may be edited.

There are a number of preferences available allowing visual customization (colors and fonts) of the IP Diagram view. These preferences are changed on the IP Diagram Preference Page (see page 199).

See also: Creating an IP Configuration (see page 306)

Note

If the selected Editor is not an IP Configuration Editor, or if the selected IP does not support a diagrammatic visualization, the IP Diagram view displays a notice that there is no diagram available for the selected Editor as shown.



IP Libraries View

The IP Libraries view provides an alternate method for creating IP configuration (`.acxip`) files versus the main menu (**File** → **New** → **IP Configuration...**). Expanding a device family name (IP Library) displays a list of available IP types for that family, double-clicking the IP type or clicking the **Create New IP Configuration** button opens the New IP Configuration Dialog (see page 166).

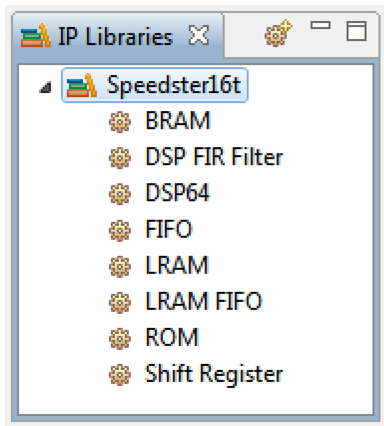


Figure 24: ***Screenshot of IP Libraries View***

> **Note**
>
> The displayed IP Libraries and IP types are dynamic and change based on which technology libraries and devices are installed and licensed. The screenshots and example descriptions in this section do not necessarily reflect the IP types of the actual device currently in use.

Table 39: ***IP Libraries Toolbar Buttons***

Icon	Description
	Opens the New IP Configuration Dialog (see page 166) to allow creating a new IP configuration file.

See also: Creating an IP Configuration (see page 306).

IP Problems View

The IP Problems view displays all of the warnings and errors for all of the currently open IP Configuration Editors.

The top half of the view displays a sorted tree table of all errors in order by IP configuration (`.acxip`) file, then all warnings in order by file. When an IP problem is selected in this tree table, further details about the problem are displayed below the tree table (in the bottom half of the view).

Double-clicking an error or warning opens the relevant IP Configuration Editor to the appropriate page (see also Creating an IP Configuration (see page 306)).

> **Note**
>
> Unlike other IP-related views, this view shows information for all open IP Configuration Editors, not just the top /active Editor.

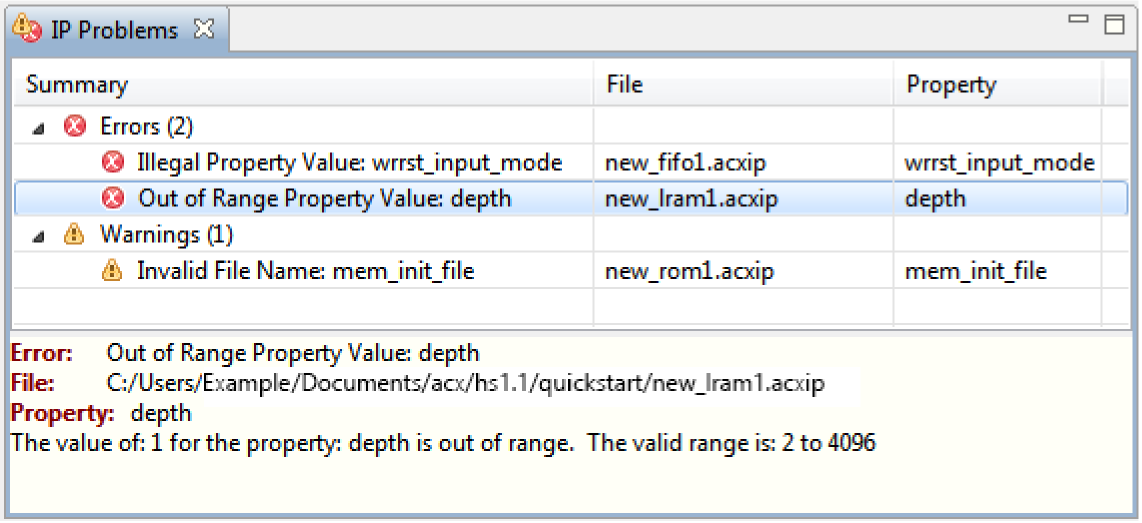


Figure 25: ***Example Screenshot of the IP Problems View***

Table 40: ***IP Problems View Icons***

Icon	Description
	Warning
	Error

Table 41: ***IP Problems View Table Columns***

Column Name	Description
Summary	A brief summary statement of the IP Configuration problem.
File	The IP Configuration file containing the error. This is the name of the file being edited in an open IP Configuration Editor.
Property	The property which is part of the IP Configuration problem. Individual properties are usually similar to the field names shown in the IP Configuration Editor. The raw properties and their values can be viewed in the IP Configuration Editor by selecting the **File Preview** tab at the bottom of each IP Configuration Editor. The **Configuration** tab shows a more user-friendly representation of the same data.

Multiprocess View

Similar to the Tcl command run_multiprocess (see page 587), the Multiprocess View () allows Running Multiple Flows in Parallel (see page 282) and Attempting Likely Optimizations Using Option Sets (see page 352).

The Multiprocess View provides a means to select multiple Implementations (see page 215) within a single project (see page 215) for flow execution. Depending upon how this view is configured, the selected implementations may be queued for sequential flow execution, run all at the same time in parallel, or (a combination) in a configurable number of parallel sequential queues. The selected implementations may be executed in the background of the workstation running ACE, or may optionally be sent to an external cloud/grid/batch job system for execution.

The Multiprocess view may also help to explore the solution space provided by various ACE optimizations. The Multiprocess View can optionally generate new implementations derived from the current Active Project and Implementation (see page 221), where each newly generated implementation applies an overlay of likely implementation option (see page 215) optimizations over the active implementation options. These collections of potentially optimized implementation options are termed option sets (see page 215).

By default, the Multiprocess View is a part of the Projects perspective (see page 24). To make the Multiprocess View visible from within any perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Multiprocess**.

This view is broken up into several sections:

- Execution Queue Management
- Multiprocess Flow Management
- Select Implementations
- Multiprocess Run Logs

Each section includes a brief descriptive paragraph describing its purpose. Each section may be collapsed and expanded by clicking the section title. Collapsing or expanding any section causes the other sections to be resized to fit the available data and view area.

For more detailed information on how to use this view, please see Running Multiple Flows in Parallel (see page 282) and Attempting Likely Optimizations Using Option Sets (see page 352).

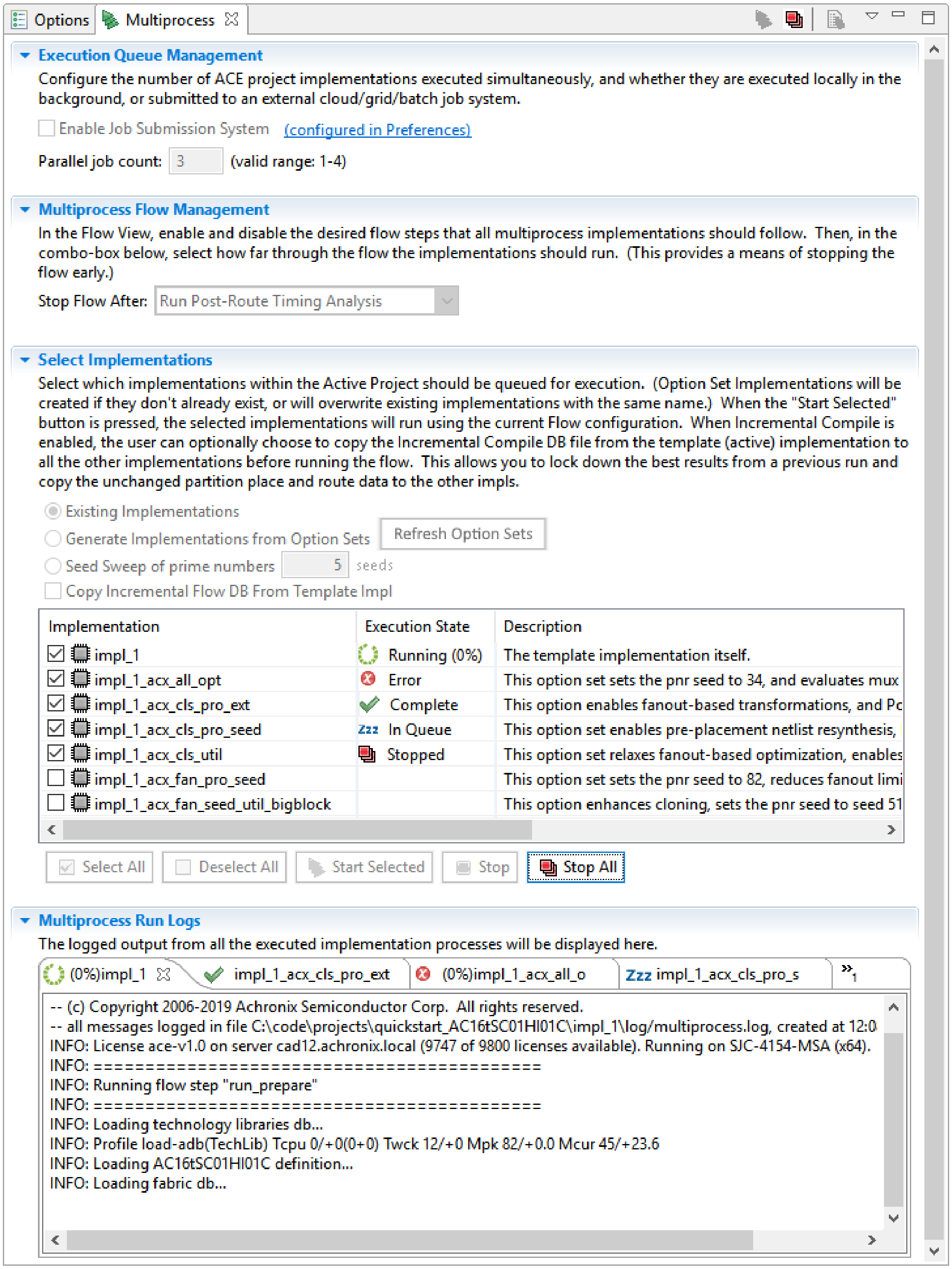


Figure 26: ***Multiprocess View Example***

Table 42: ***Multiprocess View Toolbar Buttons***

Icon	Action	Description
	Start Background Queue Execution	Starts execution of all implementations selected in the **Select Implementations** table in the number of parallel processes specified by **Parallel Queue Count**.
	Stop All Background Queue Execution	Stops/cancels execution of all currently running/queued implementations.
	Open Multiprocess Report	Opens the Multiprocess Summary Report (see page 238) for the selected project.

Execution Queue Management

This section configures the number of background processes allowed to run in parallel, and how/where they are executed.

Table 43: ***Execution Queue Management Controls***

Name	Description
Parallel Job Count	Sets the number of implementations allowed to execute in parallel. Defaults to **2**. When in background mode, the maximum allowed value is the number of available processor cores detected. When in Job Submission System mode, the maximum allowed value is 99.
Enable Job Submission System Support	When unchecked, background processes run locally on the workstation currently running the ACE GUI. When checked, ACE uses the cloud/grid/batch job submission system as configured in the preferences.
(configured in Preferences)	When selected, brings up the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201) to fully configure which cloud/grid/batch job submission system is used.

When the **Parallel Job Count** is set to the minimum value of **1**, all selected implementations are executed sequentially, one at a time. A value of **2** causes all selected implementations to be queued, and then the first two queued implementations are allowed to execute at the same time. As soon as an implementation completes its flow execution, the next queued implementation starts flow execution and the Multiprocess Summary Report (see page 238) is updated with information gathered from the just-completed implementation.

By default, ACE executes implementations in parallel by starting a background process on the host workstation for each implementation (termed "background mode"). In this case, the effectiveness of parallel implementation execution is naturally limited by the resources of the host workstation (the number of processor cores and the physical RAM).

Alternately, ACE may execute the implementations in processes distributed among multiple hosts via an external job submission system, which theoretically allows for far greater parallel compute resources. The job submissions are performed through a user-configured command line executable. This executable is configured via the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201), reached easily by following the (*configured in Preferences*) hyperlink.

See Running Multiple Flows in Parallel (see page 282) for important details regarding parallel implementation execution, configuration, and external job submission tool support.

Multiprocess Flow Management

This section allows altering how far the flow is executed for the multiprocess implementations.

Table 44: *Multiprocess Flow Management Controls*

Name	Description
Stop Flow After (1)	Allows overriding standard flow behavior and stopping the flow early — the flow step selected becomes the final flow step executed by all multiprocess implementations. Useful when steps late in the flow are known to fail with reported errors, but it is still desired to run multiple implementations through earlier parts of the flow.

Table Notes

1. The flow step chosen here is always enabled when the multiprocess run executes, regardless of whether it was enabled before the multiprocess run is launched.

See Running Multiple Flows in Parallel (see page 282) for further details regarding multiprocess flow configuration.

Select Implementations

This section allows:

- Selecting which implementations to execute (implementations derived from option sets (see page 215) are created if selected).
- Starting or stopping the execution of all selected implementations.
- Providing simple execution state feedback.

See Running Multiple Flows in Parallel (see page 282) for further details regarding selecting the implementations to be run in parallel, starting/stopping/cancelling parallel execution, etc.

See Attempting Likely Optimizations Using Option Sets (see page 352) for explanations of how to use option sets to achieve better QOR.

Table 45: *Select Implementations Controls*

Name	Description
Existing Implementations	Updates the contents of the Implementation Table to show all existing implementations for the current active project (see page 221).
Generate Implementations from Option Sets	Updates the contents of the Implementation Table to show the current active implementation (see page 221) and a number of to-be-generated implementations, one per Option Set (see page 215). The use of this radio button selection is covered in more detail at Attempting Likely Optimizations Using Option Sets (see page 352). If the Implementation Table does not show any implementations besides the active implementation while in this mode, click the **Refresh Option Sets** button.
Refresh Option Sets(1)	Causes ACE to analyze the current Active Project and Implementation (see page 221) to (re-)generate customized option sets most likely to improve QoR, then the Implementation Table is updated with a list of to-be-generated implementations.

Name	Description
Seed Sweep of prime numbers	Updates the contents of the Implementation Table to show a number of to-be-generated implementations. Each of these implementations is identical to the currently active implementation, with the implementation option "seed" being automatically set to the next consecutive prime number. The **seedcount** text field beside this radio button can be used to choose how many such implementations should be created.
Implementation Table	A table containing implementation names along with their selection state and execution state. The implementations listed vary based upon the active project and implementation (see page 221), in combination with the state of the radio buttons.
Select All	Selects all implementations in the Implementation Table.
Deselect All	Deselects all implementations listed in the Implementation Table.
Start Selected	Queues all implementations selected in the Implementation Table and begins executing in the configured number of parallel processes.
Stop All	If clicked, all currently queued implementations are removed from the queue(s) and all currently executing implementations are killed. The Multiprocess Summary Report (see page 238) is updated with any and all captured information.

Table Notes

1. This button must be clicked prior to clicking the ( ) **Start Selected** button whenever the active project or implementation has changed significantly, as well as the first time **Generate Implementations** from a new active project or implementation is chosen.

All the controls in this section center around what is in the table. The radio buttons change which implementations are listed in the table, and the push-buttons below the table change the selection state of the listed implementations, or alter the execution state of the implementations (the purpose of the entire view).

The table contents are kept in sync with the current Active Project and Implementation (see page 221). Changing active projects (which implicitly changes active implementations) updates the Implementation Table contents according to the current radio button selection.

The Implementation Table columns are each described in the following table.

Table 46: *Implementation Table Columns*

Column Name	Description
Implementation	Contains the implementation name, along with a checkbox indicating implementation selection, and an icon representing the implementation.
Execution State	Contains the execution state of the implementation.

Column Name	Description
Description	Blank when **Existing Implementations** is selected. When **Generate Implementations from Option Sets** is selected, contains a description of the Option Set (see page 215) which is used as the overlay on the active implementation (see page 221) when generating the new implementation.

Tip

If the implementation table is not large enough (or is too large) for the full implementation list, simply collapse and/or expand one of the other sections in this view (click the section title). This causes the table to resize to exactly fit the entire current implementation list.

Implementation Execution States

There are a number of possible Execution States (as listed in the second column) for the implementations in the table corresponding to the lifetime of a Multiprocess View background process. The icons from these states are also used on the tabs within the Multiprocess Run Logs section.

Table 47: *Implementation Execution States and Icons*

Icon	Execution State	Description
	blank	This implementation has not been selected for execution.
	Selected	This implementation is currently selected for execution, and execution has not been started.
Zzz	In Queue	Execution of the selected implementations has been started, this implementation was selected for execution and is currently waiting in the queue for execution.
Zzz	Scheduled	Execution of the selected implementations has been started, this implementation was selected for execution, is at the head of the queue and is being prepared for execution (this state typically only lasts for a fraction of a second).
	Running	This implementation was selected for execution, and is currently executing. Log messages should be visible in the tabbed logging area.
	Complete	This implementation was (and still is) selected for execution, and its last execution was completed without flow errors (but does not mean that the design met timing). Log messages should be visible in the tabbed logging area. Summary information should be visible in the Multiprocess Summary Report (see page 238).
	Stopped	This implementation was (and still is) selected for execution, but its last execution was stopped (possibly canceled before it even started). If its execution had started, log messages should be visible in the tabbed logging area. If Post-Route Timing Analysis or Sign-off Timing Analysis were completed for this implementation, the timing results should be visible in the Multiprocess Summary Report (see page 238).
	Error	This implementation was (and still is) selected for execution, but its last execution exited with reported errors. A tooltip for the error icon provides a summary of the detected error messages. Detailed log messages should be visible in the tabbed logging area. If Post-Route Timing Analysis or Sign-off Timing Analysis were completed for this implementation, the timing results should be visible in the Multiprocess Summary Report (see page 238).

Multiprocess Run Logs

This section shows the logs for each selected implementation as they execute. A separate tab is provided for each individual implementation. The log info is updated live as background processes execute. Depending upon configuration, external cloud/grid/batch jobs may have their log info updated live, or it may not be updated until the job is completed (the displayed log info mirrors the information captured in the log file for each implementation).

Each tab includes the name of the implementation and the execution state, which updates live. If an implementation enters the Error state, the tooltip for the tab title is updated to include a summary of the captured error messages. Error details are visible in the log shown in the tab, as well as within the Log Files (see page 218) for each implementation.

Netlist Browser View

The Netlist Browser view provides a graphical, tree-based visualization of the user design hierarchy, as found in the netlist. The displayed netlist includes the results of any transformation, legalization, etc. that have happened through the current stage in the Flow.

For large designs, there are a tremendous number of objects in the netlist. To simplify the view, the Netlist Browser provides a number of ways to filter the flood of data down to just the most useful information (there are no filters active by default).

Each instance node in the tree includes the instance name and the cell type. Macros include the macro name, and the counts of the various major logic types contained within that macro. Be aware that these logic type counts are not affected by the filters; the numbers shown always represent the unfiltered total counts. Clock domain names and Partitions names also are listed when appropriate.

By default, the Netlist Browser view is included in the Floorplanner perspective (see page 24). To add it to other perspectives, select **Window** → **Show View...** → **Other...** → **Achronix** → **Netlist Browser**.

As can be seen in the second column of the example below, three instances have been highlighted:

- `inb1_ibuf[3].x_ipad_i_io_buff` in cyan
- All members of the `z0_obuf.*` macro hierarchy in pink
- `inb1_int_z[2]` in dark blue

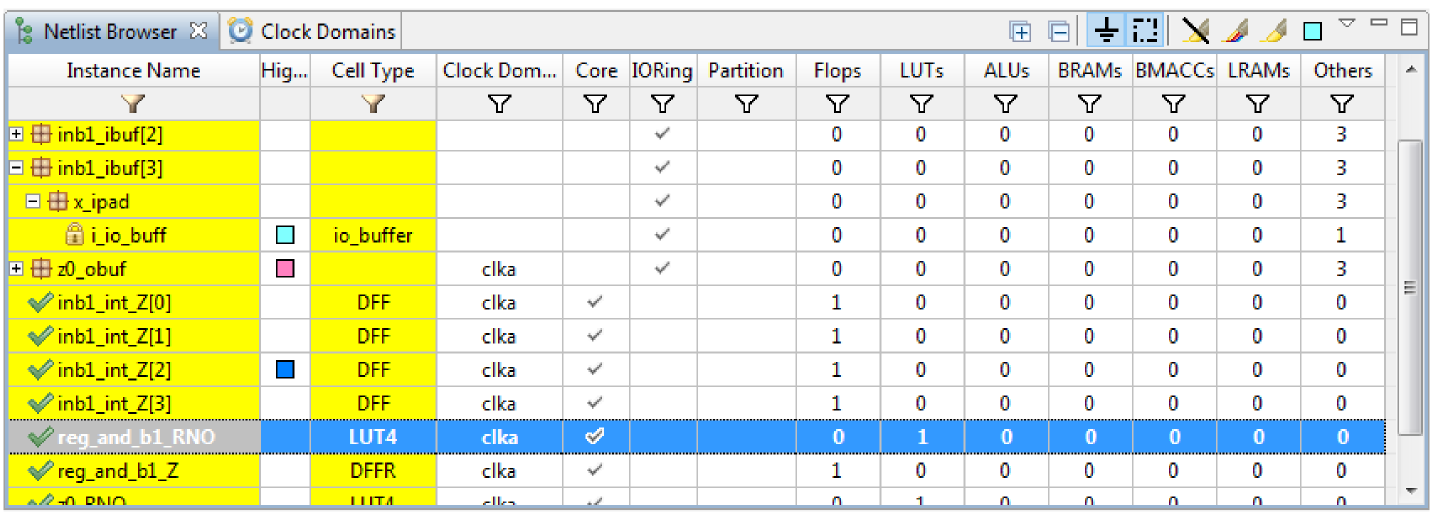


Figure 27: *Netlist Browser Example*

Note

- The small colored square in the toolbar shows the active highlighting color. If highlighting is applied to a macro then all “child” instances within are also set to the current highlight color.
- Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The screenshots and example descriptions in this section do not reflect the resource types of actual devices.

Table 48: ***Netlist Browser Table Columns***

Column Name	Description
Instance Name	The name of the instances in the netlist. Instances within a macro are grouped together as leaves under the macro branch. Additionally, an icon is used to indicate the placement state of the instance. The possible icons are shown in a separate table below.
Highlight Color	• For instances, shows a color square to indicate the instance highlight color, if any. • For macros, if all contained instances have the same highlight color, the macro shows a color square for that same highlight color. If even one contained instance has a different highlight color, or no highlight at all, the macro displays no color square. This value does not change for macros during filtering.
Cell Type	• For instances, shows the cell type of the instance. • For macros, this column is blank.
Clock Domain	• For instances, shows a list of all the clock domains of which the instance is a member. • For macros, shows a summary list of the clock domains for all the contained instances. This value does not change for macros during filtering.
Core	• For instances, this is checked if the instance is considered a member of the Core, or blank if it is not. • For macros, this is checked if any contained instances are considered a member of the Core, or blank if no contained instances are in the Core. This value does not change for macros during filtering.
IORing	• For instances, this is checked if the instance is considered a member of the IORing, or blank if it is not. • For macros, this is checked if any contained instances are considered a member of the IORing, or blank if no contained instances are in the IORing. This value does not change for macros during filtering.
Partition	For both instances and macros, the name of the Partition to which the item belongs, if any. See Using Incremental Compilation (Partitions) (see page 363)
Resource	• For instances, this is one if the instance is of type *resource*, or zero otherwise. • For macros, this is the sum count of all contained *resource* instances (regardless of filtering).

Icons decorate all the nodes in the tree in the Instance Name column.

Table 49: *Netlist Browser View Icons*

Icon	Description
	Macro
	Unplaced Instance
	Placed Instance (Soft)
	Placed Instance (Fixed)

A number of actions are available in the view via:

- Buttons at the top of the view
- The (...) ellipsis view menu button
- Right-click context menus on the nodes of the tree

> **Note**
>
> If these actions are performed upon macros, all child leaf nodes, even those currently filtered to be hidden in the tree, are affected by the chosen action.

Table 50: *Netlist Browser View Actions*

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Add to Selection		Y		Adds the item(s) to the ACE Selection Set (as shown in the Selection View (see page 136)).
	Remove from Selection		Y		Removes the item(s) from the ACE Selection Set (as shown in the Selection View (see page 136)).
	Choose Highlight Color	Y	Y		Determines which color is applied to the objects chosen from the tree the next time the Highlight action is selected for this view.
	Highlight	Y	Y		Applies the currently active Highlight color to the chosen item(s) in the tree. See Highlighting Objects in the Floorplanner View (see page 315).
	Un-Highlight	Y	Y		Clears the Highlight for the chosen item(s) in the tree. When painted in the Floorplanner view, the chosen item(s) now use their default color(s) instead of a highlight color.
	Auto-Highlight	Y			Automatically applies unique highlight colors to all visible core hierarchy levels in the tree. IORing hierarchy levels are skipped.
	Zoom To		Y		Zooms the Floorplanner view to a region containing the instances currently chosen in the tree.
	Show in Netlist (1)		Y		Attempts to open a text editor to the file and line number relevant to the chosen instance. Available only when a single instance is chosen in the view.

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Unfix Placement of Instance		Y		Changes the state of an already-placed instance from Fixed Placement to Soft Placement. This choice is only available when an instance already has Fixed Placement.
	Fix Placement of Instance		Y		Changes the state of an already-placed instance from Soft Placement to Fixed Placement. This choice is only available when an instance already has Soft Placement.
	Unplace Instance		Y		Completely removes the site assignment for an instance, making it Unplaced. This choice is only available when an instance is already Placed.
	Expand All	Y			Expands all collapsed macro branches in the tree, making all leaf instances visible.
	Collapse All	Y			Collapses all expanded macro branches in the tree.
	Show Power and Grounds	Y		Y	Disabled by default. When disabled, all instances of power and ground are hidden within the tree, effectively acting as a filter. Therefore, the **Cell Type** column gains the active filter indicator (yellow background by default) when power and ground are hidden. This filter is a higher priority than user custom filters applied to the **Cell Type** column (if a custom filter is created to expose only "GND" grounds, but **Show Power and Grounds** is disabled, the "GND" instances remain hidden).
	Show Boundary Pins	Y		Y	Disabled by default. When disabled, all instances of boundary pins are hidden within the tree, effectively acting as a filter. Therefore, the **Cell Type** column gains the active filter indicator (yellow background by default) when boundary pins are hidden. This filter is a higher priority than user custom filters applied to the **Cell Type** column (if a custom filter is created to expose only "OPIN" instances, but **Show Boundary Pins** is disabled, the "OPIN" instances remain hidden).
	Show Feedthrough LUTs (2)			Y	Toggles the visibility of all feedthrough instances, most often created by Achronix optimizations. These always have "_ft_" plus some additional notation in the (ACE-generated) instance name. Enabled by default. When disabled, all feedthrough instances (might be more than just LUTs in some cases) are hidden within the tree, effectively acting as a filter. Therefore, the **Instance Name** column gains the active filter indicator (yellow background by default) when feedthroughs are hidden. This filter is a higher priority than the user custom filters applied to the **Instance Name** column (if a custom filter is created to expose only instances that contain "LUT" in the name, but **Show Feedthrough LUTs** is disabled, any instances that have a feedthrough-based name remain hidden, even if they have an explicit "LUT" in the name).
	Show Constants (2)			Y	Toggle the visibility of all instances with "const" somewhere in the name, most often created by Achronix optimizations. Enabled by default. When disabled, all instances with "const" anywhere in the name are hidden within the tree, effectively acting as a filter. Therefore, the **Instance Name** column gains the active filter indicator (yellow background by default) when constants are hidden. This filter is a higher priority than the user custom filters applied to the **Instance Name** column (if a custom filter is created to expose only instances that contain "LUT" in the name, but **Show Constants** is disabled, any instances that have a "const" in their name remain hidden, even if they have an explicit "LUT" in the name).

Table 51: *Netlist Browser View Actions (cont.)*

	Show Duplicates /Clones (2)			Y	Toggle the visibility of instances with various forms of "_DUP_" in the name, most often created by Achronix optimizations. Enabled by default. When disabled, instances with "_DUP_" or "_dup_" in the name (typically with some additional notation) are hidden within the tree, effectively acting as a filter. Therefore, the **Instance Name** column gains the active filter indicator (yellow background by default) when duplicates/clones are hidden. This filter is a higher priority than the user custom filters applied to the **Instance Name** column (if a custom filter is created to expose only instances that contain "LUT" in the name, but **Show Duplicates/Clones** is disabled, any instances that have a "_DUP_" in their name remain hidden, even if they have an explicit "LUT" in the name).
	Toggle Filter Row Visibility (3)	Y	Y		Changes whether the filter row (of filter icons) is visible or not.
	Configure view...		Y		Jumps to the Netlist Browser view in the Preferences dialog.

Table Notes

1. **Caution:** This is Early Access functionality and might not always open the text editor to the expected location.
2. **Caution:** This filter uses an imperfect name-string-matching heuristic and might, in rare cases, intercept user instances as false-positives.
3. Toggle row visibility does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Warning!

- Be aware that when actions are performed upon macros, all the children of that macro, even the invisible /filtered nodes, are affected.
- With default preference settings, in the Floorplanner View (see page 53), Highlight colors of (placed) instances are only visible when the Instances Layer is enabled, and the instances are not members of the ACE Selection Set. This is because the Instance Selection color has a higher priority than the Highlight color.

Filtering Displayed Instances

Some convenience filters for **Cell Type** are already present as visibility toggles in the Netlist Browser. These filters are represented by the ( ) **Show Boundary Pins** and ( ) **Show Power and Grounds** toggle actions/buttons. Be aware that the "hide" functionality of these actions (when the Show toggle is disabled) is considered a higher-priority filter than any user custom filters. For example, when boundary pins are hidden due to this toggle, even if a custom filter tries to expose boundary pins, the toggled filter wins, and the pins remain hidden.

Additionally, some convenience filters for **Instance Name** are already present as visibility toggles. These filters are represented by the **Show Feedthrough Luts**, **Show Constants**, and **Show Duplicates/Clones** toggled menu items. Be aware that the "hide" functionality of these actions (when the Show toggle is disabled) is considered a higher-priority filter than any user custom filters. For example, when Constant instances are hidden due to this toggle, even if a custom filter tries to expose constants, the toggled filter wins, and the constant instances remain hidden.

To enable custom instance filter manipulations, it might be necessary to click ( ) **Toggle Filter Row Visibility** to cause the filter manipulation row to become visible. This toggle action is available in a context menu when right-clicking any table column header and is also available in the view supplemental menu (the small down arrow icon in the upper-right of the view, to the left of the **Minimize View** button).

Most columns of the table can filter the displayed instances (not the macros) by value. When filtering by column value, only instances with column values matching the filter are retained; non-matching values are excluded from the table.

Be aware that macro rows do not directly respond to filters, and remain visible as long as any single child instance remains visible. When all child instances of a macro are hidden, the parent macro is hidden as well. On a related note, macro summary counts in numeric columns (as when counting LUTs in a macro) do not change when filters are applied. The displayed counts are always the complete, unfiltered counts.

Warning!

- When using filters, the values being filtered are those of the individual instances, not the macros. Macros are filtered out only if all of their children are filtered out. As a result, when filtering by the logic types, the only possible filter numeric values in this table are 0 or 1, because these are the only legal values for an instance.
- Also, be aware that when filtering the **Instance Name** column, the parent macro names are considered part of the instance name — the prefix (the fully qualified instance name is used, not just the terminating leaf name).

Columns containing text can be filtered by string value (simple wildcard substring matching by default, but Regular Expression matching using Java rules is also available, see https://en.wikipedia.org/wiki/Regular_expression). Columns with checkmarks can be filtered by Boolean value. Columns containing numbers can be filtered by numerical value.

To add a filter to a column:

1. Click the ( ) filter icon, which causes a data-appropriate filter dialog to appear.
2. Fill in the desired filter values.
3. Click **Apply** to apply the filter to the instances in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the ( ) active filter icon.

An example filter for the **Cell Types** column that uses Regular Expressions to block PWR, GND, ASC, SAC, and bit* cell types is shown below.

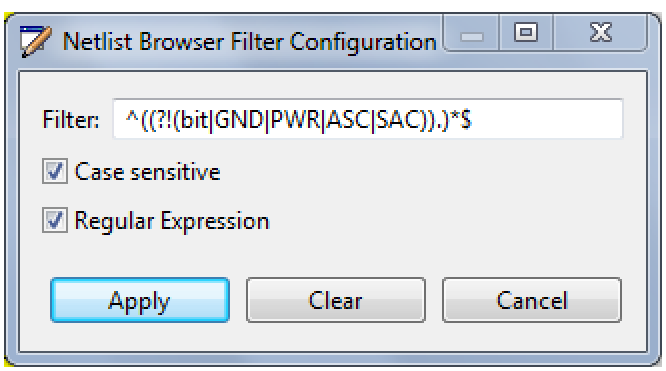


Figure 28: ***Cell Types Column Filter Example***

To edit (or clear) an existing filter:

1. Click the ( ) active filter icon causing the data-appropriate filter dialog to appear, this time pre-populated with the existing filter setting.

2. Change the filter value and click **Apply** again to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the ( ) inactive version.

Drag-and-Drop

The Netlist Browser supports a limited set of Drag-and-Drop interactions with other views in the Floorplanner perspective (see page 24). The Netlist Browser view only acts as a Drag-and-Drop source; items dropped on the Netlist Browser view are ignored.

Any node of the tree may be dragged to the Tcl Console view (see page 144), and when dropped anywhere in the view, appropriate text is inserted at the beginning of the Tcl command-line.

Instance nodes may also be dragged to the Floorplanner view (see page 53). When dropped on the Floorplanner view, the behavior depends upon the current Tool mode. When the Floorplanner **Placement/Panning Tool** is active, placement is attempted.

Any node of the tree may be dragged to the Placement Regions view (see page 120) or the Floorplanner view (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 358). Dragging a macro is the equivalent of dragging all individual instances which are members of that macro.

NoC Performance View

The NoC Performance view shows an interactive diagram that includes the I/O ring and the 2D NoC resources for the target device. Loading a simulation log file produced by the device simulation model (DSM) into the view provides graphic visualization of traffic between Network Access Points (NAPs) for different periods of time during the simulation ("time slices").

Loading Simulation Log Files

Load simulation log files by clicking **Browse** on the view control bar as shown:

Figure 29: ***Simulation Log File Loading Example***

When a log file has been loaded, hover the mouse pointer over the **Log** text field to see the full path to the loaded file, as well as the name of the target device on which the simulation was running:

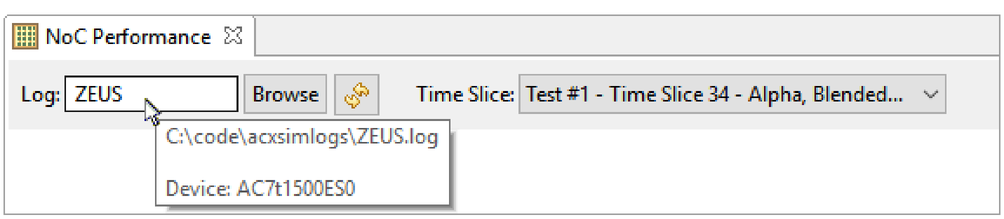


Figure 30: ***Simulation Log File Path Example***

Browsing Time Slices

Use the Time Slice combo box on the view control bar to choose a time slice to visualize. The statistics in the chosen time slice are used to colorize portions of the view diagram.

There is a "tool box" of additional controls in a collapsible flyout panel to the right of the diagram, similar to the one found in the Floorplanner View (see page 53):

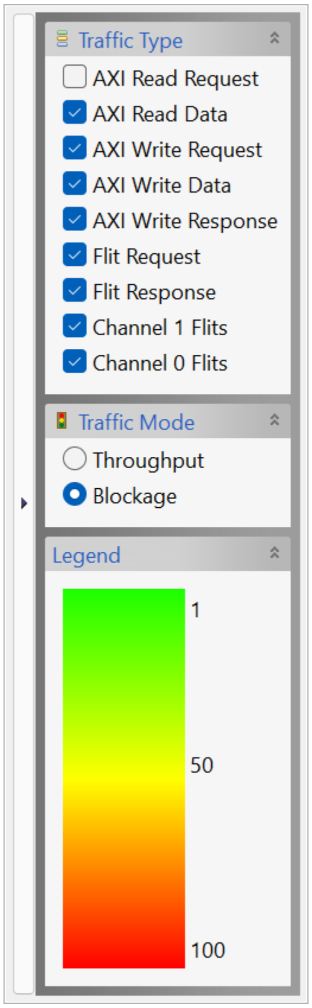


Figure 31: *Time Slice Tool Box Panel Example*

The tool box contains three main sections:

- Traffic Type – toggles that allow choosing which types of traffic data to display in the diagram.
- Traffic Mode – the diagram can be switched between two display modes: **Throughput** or **Blockage**.
- Legend – displays the gradient range coloring for the selected Traffic Mode.

In Throughput mode, the diagram is colored using the following gradient range:

Table 52: *Throughput Mode Gradient Range*

Throughput Gradient	Default Color
High	Green
Medium	Light blue
Low	Darker blue

The Throughput mode diagram coloring is illustrated below:

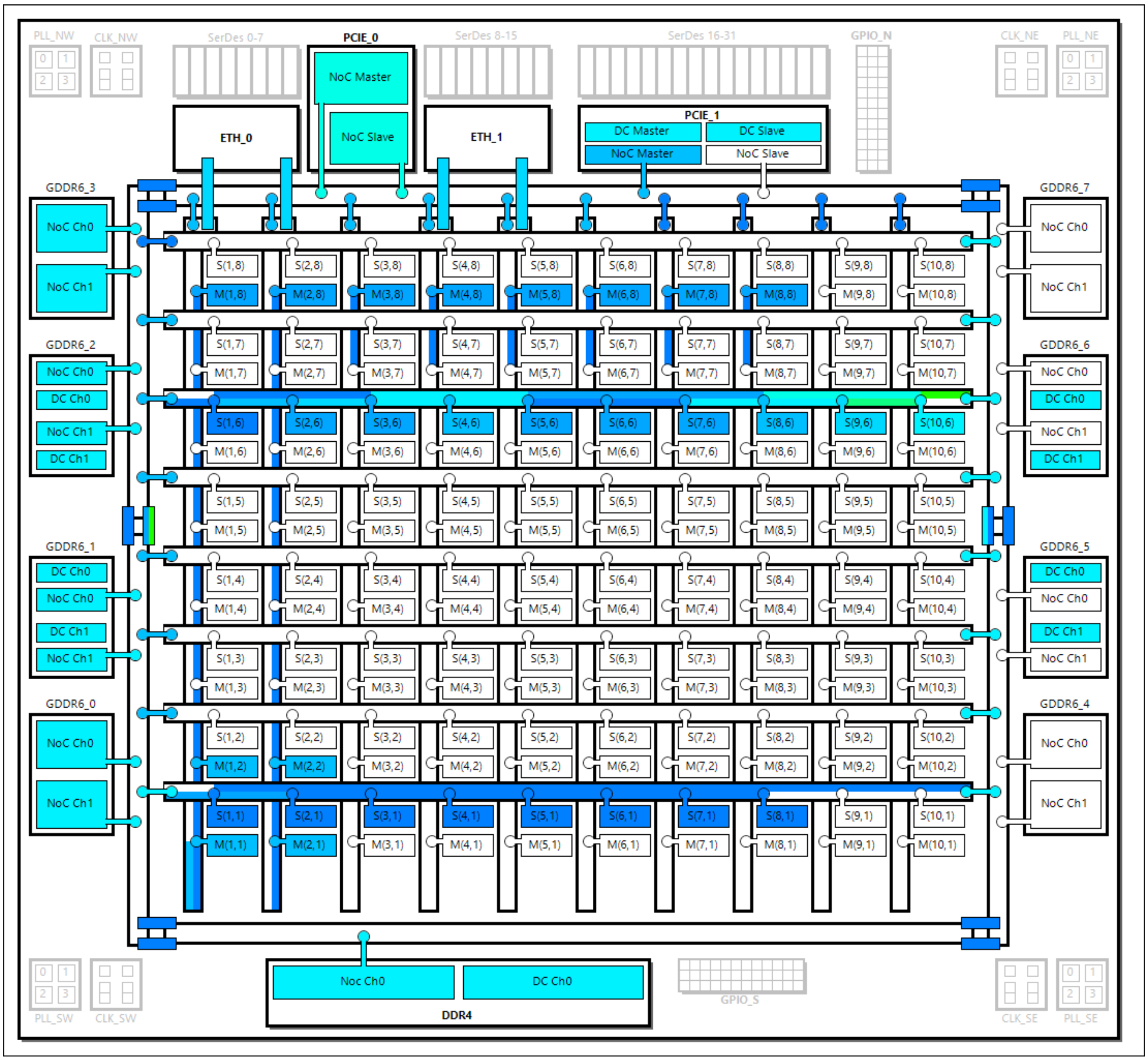


Figure 32: *Throughput Mode Diagram Coloring Example*

In Blockage mode, the diagram is colored using the following gradient range:

Table 53: *Blockage Mode Gradient Range*

Blockage Gradient	Default Color
Low	Green
Medium	Yellow
High	Red

The Blockage mode diagram coloring is illustrated below:

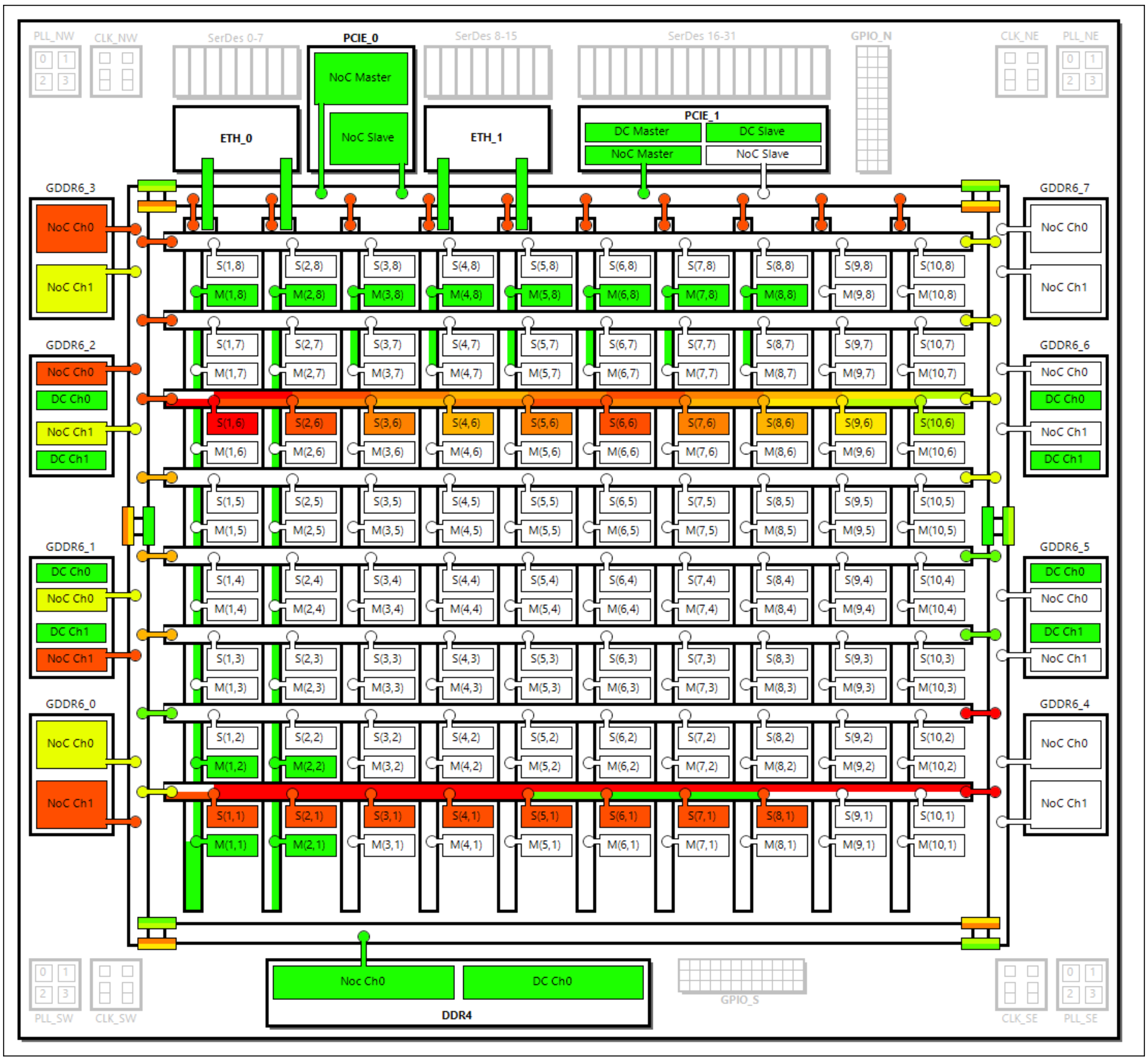


Figure 33: *Blockage Mode Diagram Coloring Example*

Hover the mouse pointer over any colored portion of the diagram and a tool tip is displayed showing the raw data from the simulation log file used to determine the color.

There is also a breakdown of how much time was spent "idle" versus "trying," with the "trying" time further broken down into time spent "blocked" and time spent "transferring":

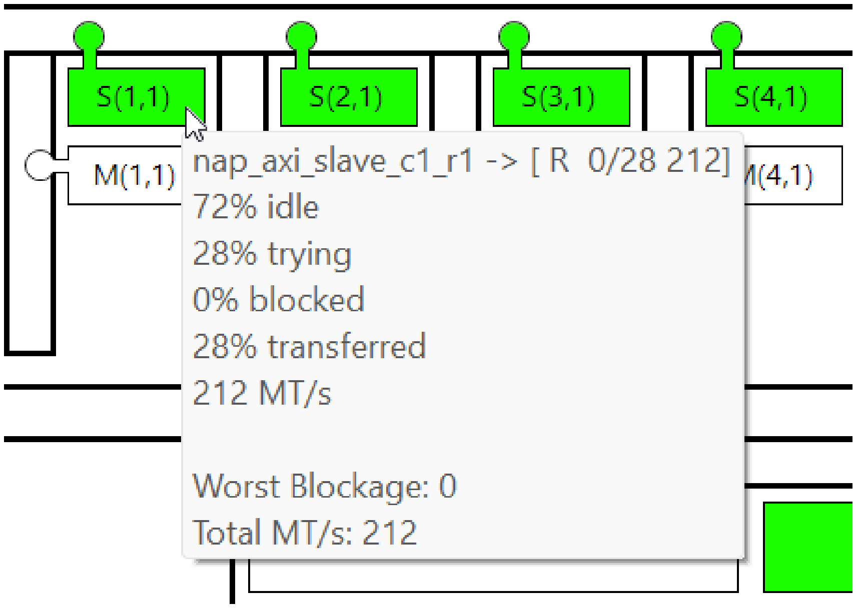


Figure 34: *Simulation Data Tool Tip Example*

Zooming

The view can be zoomed in or out several levels with the mouse wheel while the mouse pointer is hovering over the diagram.

Drag-Scrolling

When zoomed in, the diagram can be scrolled in any direction by clicking and dragging the mouse pointer anywhere inside of the diagram.

Adjusting Diagram Properties

The font used in the diagram, as well as the colors used to generate the throughput and blockage gradients, can be configured in the Preferences dialog.

The Preferences dialog can be accessed through the main application menu (**Window** → **Preferences**), or by clicking the "down arrow" shortcut in the upper right of the view and clicking **Configure view...**:

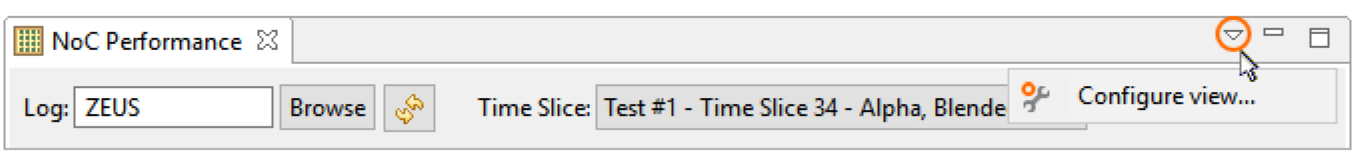


Figure 35: *Preferences Dialog Shortcut Location*

The Preferences Dialog and the available settings are shown below.

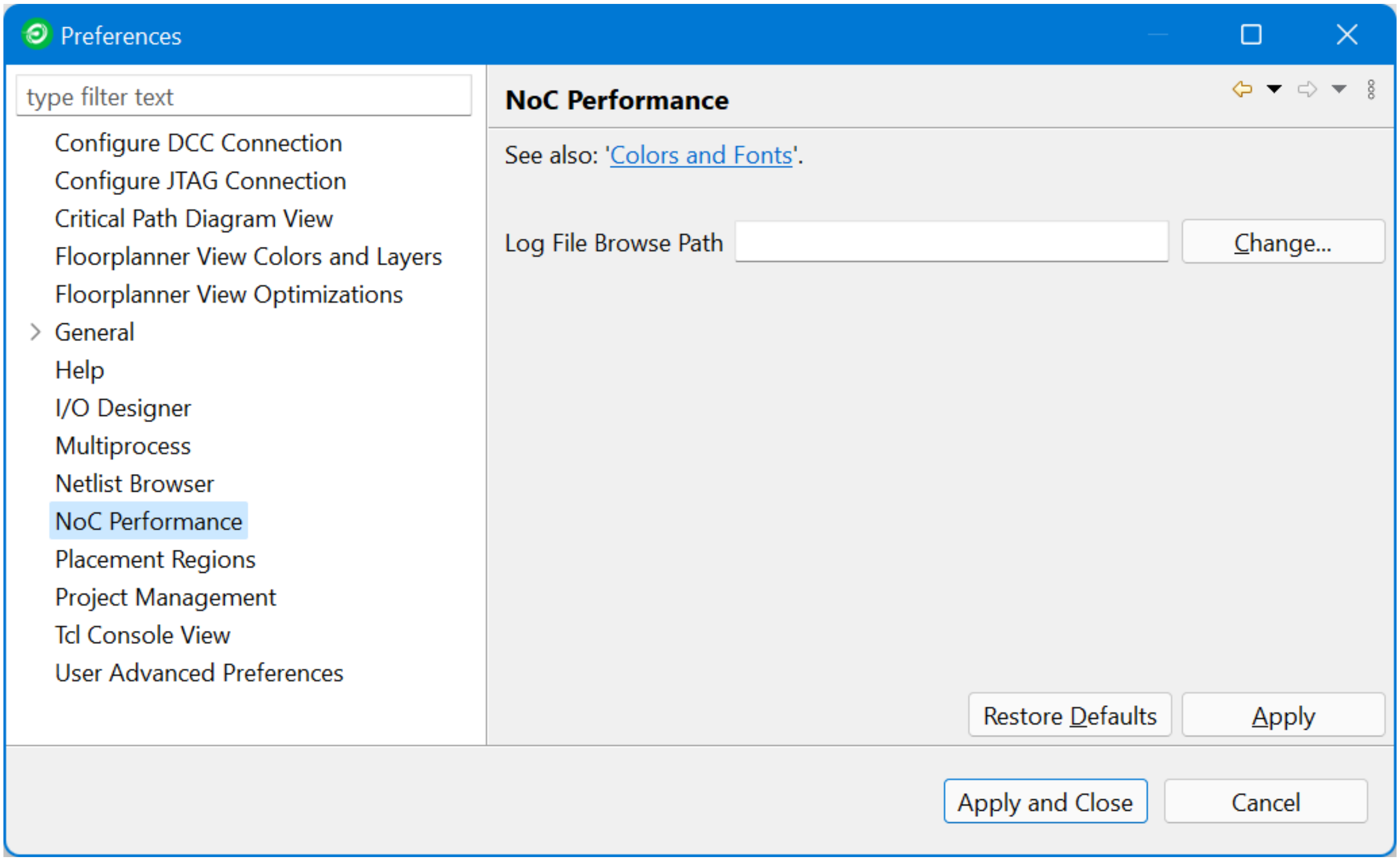


Figure 36: *NoC Performance Preferences*

Table 54: *NoC Performance Preferences*

Preference	Description
See Also: 'Colors and Fonts'	Clicking this hyperlink moves to the NoC Performance section of the **Colors and Fonts** preferences area.
Log File Browse Path	By default, the Log File **Browse** button in the NoC Performance view begins browsing at the last path used. To always start browsing in a specific folder, enter that path here.

NoC Time Slice View

The NoC Performance time slice view displays information about the time slice currently selected in the NoC Performance View (see page 95).

Statistics and "Notes" from the simulation log data can be rendered in different colors that can be specified in the application preferences.

Word-wrap can be toggled via a button in the upper-right of the view tool bar.

As with all views in ACE, this view can be dragged and dropped to any convenient location: left of the NoC Performance View, below it, etc.

NoC Time Slice

```
@123     acx_perf> AXI Read Requests, Responses
@1234    acx_perf> nap_axi_slave_c1_r1  -->  [AR 95/ 5  38] [ R  0/28 212]
@1234    acx_perf> nap_axi_slave_c2_r1  -->  [AR 95/ 5  38] [ R  0/28 212]
@1234    acx_perf> nap_axi_slave_c3_r1  -->  [AR 95/ 5  38] [ R  0/28 212]
@1234    acx_perf> nap_axi_slave_c4_r1  -->  [AR 95/ 5  38] [ R  0/28 210]
@1234    acx_perf> nap_axi_slave_c5_r1  -->  [AR 95/ 5  38] [ R  0/28 208]
@1234    acx_perf> nap_axi_slave_c6_r1  -->  [AR 95/ 5  37] [ R  0/28 207]
@1234    acx_perf> nap_axi_slave_c7_r1  -->  [AR 95/ 5  38] [ R  0/28 208]
@1234    acx_perf> nap_axi_slave_c8_r1  -->  [AR 95/ 5  36] [ R  0/26 198]
@1234    acx_perf> fabric_axi_w_r1      -->  [AR 87/13 251] [ R  0/84 1686]
@1234    acx_perf> AXI Write Requests/Data/Responses
@1234    acx_perf> ddr4_dc0_axi            -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_1_dc0_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_1_dc1_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_2_dc0_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_2_dc1_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_5_dc0_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_5_dc1_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_6_dc0_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> gddr6_6_dc1_axi         -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> pcie_x16_dc_slave_axi   -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> pcie_x16_dc_master_axi  -->  [AW  0/ 9  90] [ W  0/72 721] [ B  0/ 9  90]
@1234    acx_perf> Flit Requests/Responses
@1234    acx_perf> noc_ws_to_wn  -->  [Rq  10/18  10] [Rs  33/ 2  100]
@1234    acx_perf> noc_wn_to_ws  -->  [Rq  50/18  25] [Rs  75/ 2   80]
@1234    acx_perf> noc_wn_to_n   -->  [Rq  10/18  75] [Rs  33/ 2   60]
@1234    acx_perf> noc_n_to_wn   -->  [Rq  50/18 100] [Rs  75/ 2   40]
@1234    acx_perf> noc_en_to_n   -->  [Rq  10/18  75] [Rs  33/ 2   60]
@1234    acx_perf> noc_n_to_en   -->  [Rq  50/18 100] [Rs  75/ 2   40]
@1234    acx_perf> noc_es_to_en  -->  [Rq  10/18  10] [Rs  33/ 2  100]
@1234    acx_perf> noc_en_to_es  -->  [Rq  50/18  25] [Rs  75/ 2   80]
@1234    acx_perf> noc_es_to_s   -->  [Rq  10/18  75] [Rs  33/ 2   60]
@1234    acx_perf> noc_s_to_es   -->  [Rq  50/18 100] [Rs  75/ 2   40]
@1234    acx_perf> noc_ws_to_s   -->  [Rq  10/18  75] [Rs  33/ 2   60]
@1234    acx_perf> noc_s_to_ws   -->  [Rq  50/18 100] [Rs  75/ 2   40]
@1234    acx_perf> noc_es_to_s   -->  [Rq  10/18  75] [Rs  33/ 2   60]
@1234    acx_perf> noc_s_to_es   -->  [Rq  50/18 100] [Rs  75/ 2   40]
@1234    acx_perf> Channel 0/1 Flits
@1234    acx_perf> ethernet_flit_s_c1  -->  [C0  0/37 745]
@1234    acx_perf> ethernet_flit_n_c2  -->  [C0  0/37 745]
@1234    acx_perf> ethernet_flit_s_c2  -->  [C0  0/37 745]
@1234    acx_perf> ethernet_flit_n_c4  -->  [C0  0/37 745]
@1234    acx_perf> ethernet_flit_s_c4  -->  [C0  0/37 745]
@1234    acx_perf> ethernet_flit_s_c1  -->  [C1  10/100 1000]
@1234    acx_perf> ethernet_flit_n_c2  -->  [C1  20/100 1000]
@1234    acx_perf> ethernet_flit_s_c2  -->  [C1  30/100 1000]
@1234    acx_perf> ethernet_flit_n_c4  -->  [C1  40/100 1000]
@1234    acx_perf> ethernet_flit_s_c4  -->  [C1  50/100 1000]
```

Figure 37: ***NoC Performance Time Slice View Example***

Adjusting View Properties

The font used in the Time Slice View, as well as the colors used to differentiate statistics from notes, can be configured in the Preferences dialog.

The Preferences dialog can be accessed through the main application menu, **Window** → **Preferences**. The dialog can also be opened by double-clicking the gradient **Legend** section in the NoC Performance View toolbox.

The Preferences Dialog and the available settings are shown in the following example:

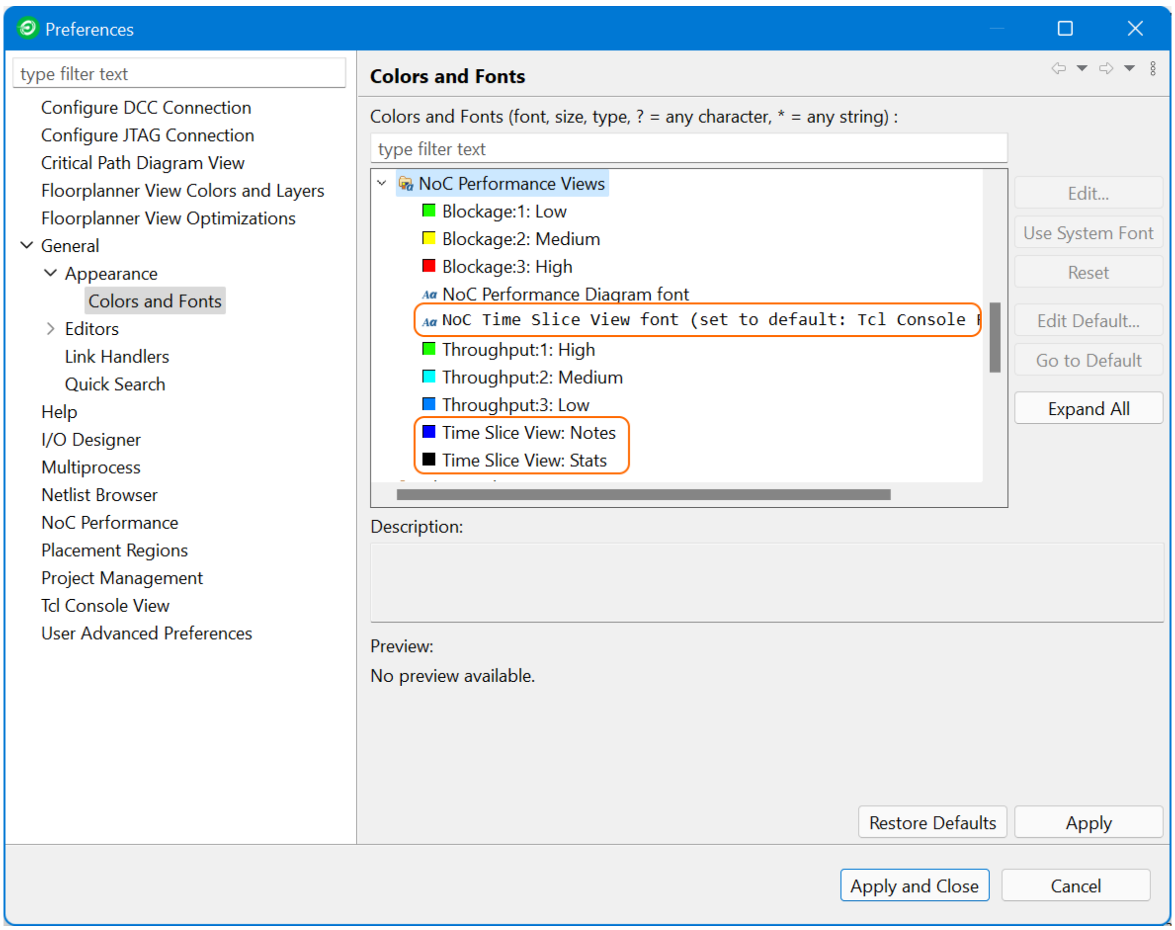


Figure 38: ***NoC Time Slice View Colors and Fonts Preferences***

Table 55: ***NoC Performance Preferences***

Preference	Description
NoC Time Slice View font	The font to be used in the NoC Time Slice view.
Time Slice View: Notes	The color used to render "notes" in the NoC Time Slice view.
Time Slice View: Stats	Color used to render "statistics" in the NoC Time Slice view.

Options View

The Options view displays project (see page 215) implementation options (see page 215) for the active implementation (see page 215). From this view, the active project implementation (see page 221) can be configured for its run through the flow (see page 221).

This view does not display any information unless an active implementation is selected in the Projects View (see page 125). When Running the Flow (see page 280), the implementation options of the active implementation are used to govern the flow.

By default, the Options view is included in the Projects perspective (see page 24). To add the Options view to the current perspective, select **Window** → **Show View...** → **Options**.

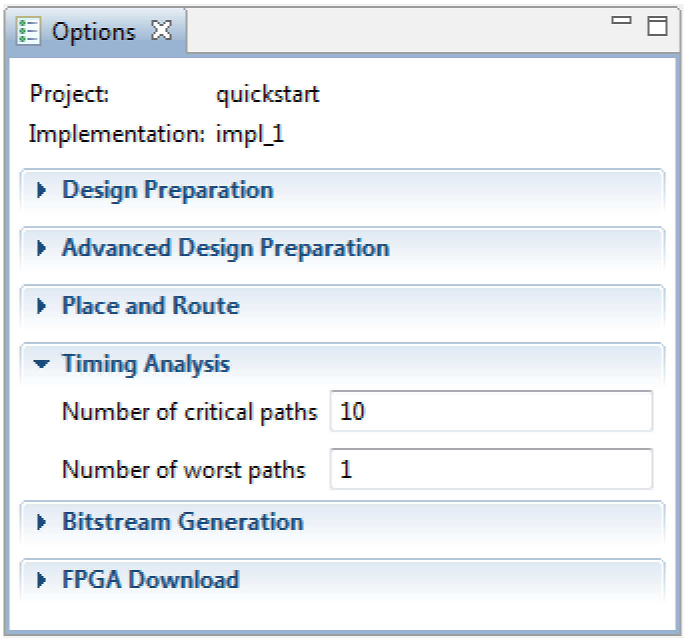


Figure 39: *Options View Example*

Tip

Tcl Equivalence
Each implementation option that can be configured via this graphical view may also be configured via the `set_impl_option` Tcl command. The current value of each option can be retrieved with the `get_impl_option` Tcl command. The values of options may be reset back to their default values with the `reset_impl_option` Tcl command.

Note

The Options view does not show all available options.

- The implementation options included in this view while ACE is running are the most used standard supported options, but are only a subset of all options available.
- Power users of ACE may also configure the Options View to display all "advanced" implementation options by setting the GUI preference under the main menu: **Window** → **Preferences** → **User Advanced Preferences** → **Display Advanced Impl Options**.
- The implementation options shown in the tables below are the subset of non-advanced options in the view that are relevant to all libraries/devices. Library-specific or device-specific options are not listed within these tables.
- A complete list (with descriptions) of all available library-specific, device-specific, and advanced implementation options, along with default values and current values, is available in the Implementation Options Report (see page 240), which can be generated with the `report_impl_options` Tcl command.

Table 56: ***Design Preparation Implementation Options***

Option	Tcl Option	Description
Target Device (1)	`partname`	Specifies the name of the FPGA part for this implementation.
Package	`package`	Specifies the FPGA package for the target device.
Speed Grade	`speed_grade`	Selects the desired speed grade for the target device.
Core Voltage	`core_voltage`	Selects the core voltage for the target device.
Junction Temperature	`junction_temperature`	Selects the junction temperature for the target device.
Flow Mode (2)	`flow_mode`	**Evaluation** mode – ignores non-fatal DRCs as long as possible, allows I /O Virtualization, and ignores missing SDC constraints to get a post-route timing report quickly. **Normal** mode – enforces all DRC checks necessary to generate a correct bitstream. Some checks are flagged as warnings early on in the flow to provide an opportunity to fix the problems (for example, fixing the placement of I/Os). These same checks may change to report an error during final DRC checks. **Strict** mode – similar to **Normal** flow mode, but enforces all DRC checks and errors out as early in the flow as possible. See also: Flow Mode (see page 226)
Auto-Select Top Module	`autoselect_top_module`	Specifies whether the top module name for this implementation should be automatically selected. A value of **1** automatically selects the name. A value of **0** forces use of the `-top_module` implementation option value.
Top Module Name	`top_module`	Specifies the top module name for this implementation when the `-autoselect_top_module` implementation option is set to **0**.
Constraint Files (3)	Uses the `enable_project_constraints` and `disable_project_constraints` commands	Enables or disables the use of an existing project SDC/PDC constraint file for this implementation flow. All constraint files defined for the active project are listed.

Option	Tcl Option	Description

Table Notes

1. **Caution:** The Chosen "Target Device" Affects Other Implementation Options.

   Each target device can have unique implementation options available within ACE, and may even have different default values for those implementation options which are shared or common between devices.

   Changing the target device value may have a ripple effect upon other option values. Thus, reviewing the values of all other options after changing the target device value may be necessary.
2. Bitstream generation requires **Normal** or **Strict** flow mode.
3. Constraint files are loaded in the order listed. To change the order constraint files are loaded, see Adding Source Files (see page 272).

Table 57: ***Advanced Design Preparation Implementation Options***

Option	Tcl Option	Description
Timing-Driven Clustering	`timing_driven_clustering`	Specifies whether timing-driven clustering is enabled during placement.
Fanout Control	`fanout_control`	When fanout control is enabled, nets with a fanout higher than the **Fanout Limit** are refactored.
Fanout Limit	`fanout_limit`	Specifies the maximum fanout any net can have when **Fanout Control** is enabled.
Fanout Limit for Critical Nets	`critical_fanout_limit`	Specifies the maximum fanout critical nets can have when **Fanout Control** is enabled.
Resynthesis Mode	`synthesis_remap`	Specifies whether resynthesis should optimize for timing, area, or should be disabled. **Off** disables all resynthesis. **Optimize for Area** can be used to reduce the total number of LUTs. **Optimize for Timing** can be used to improve timing, but may increase area. The optimizations performed depend upon the strategies chosen below. If the “Place and Route” Implementation Option **Timing-Driven PnR** is disabled, resynthesis timing optimizations are also disabled.

Table 58: *Advanced Design Preparation Implementation Options (cont.)*

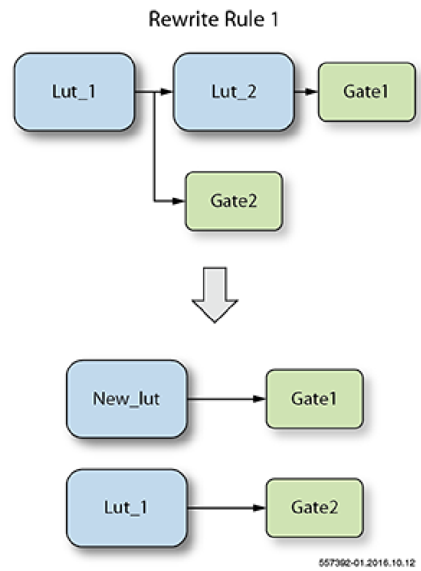
Option	Tcl Option	Description
Rewrite Rule-1	`resynthesis_rewrite_rule1`	When enabled, and **Resynthesis Mode** is set to **Optimize for Timing**, attempts to reduce the number of LUTs in series. In critical paths, Rewrite looks at the LUT programs and the number of used inputs to determine where to apply the transformation. The following transformation is then applied when feasible: 
Move Flip-flop Reset	`resynthesis_move_ff_reset`	Specifies whether resynthesis moves flip-flop reset logic to LUTs when **Resynthesis Mode** is **Optimize for Timing**.
Period of Anti-Aging Oscillator (in ns) (1)	`areafill_clock_period`	Period of areafill oscillator in ns (0 to disable). For anti-aging purposes, setting this option to a **non-zero** value causes ACE to automatically insert logic during Run Prepare to fill the area in the core fabric not consumed by the user design logic, driven by a ring oscillator which toggles at the specified period in nanoseconds. When **0**, disables insertion of anti-aging area fill logic.
Limit Anti-Aging to Clocks Paths	`anti_aging_onlyclock`	Use anti-aging areafill only on clock nodes. Data paths are not filled.
Virtual IO Style	`virtual_io_style`	Reduces the number of I/O pads by collapsing bussed ports. Only applies in Evaluation flow mode when I/O pad utilization exceeds the value of **Virtual IO Utilization**. **Off** disables this feature. **Stubout using Floating LUTs** converts the pads into unconnected LUTs. **Serialize using LUTs** reduces the bus into a single pad feeding a scan chain made of LUTs, with a second pad used to select between "load" and "shift" modes. **Serialize using DFFs** builds the scan chain from DFFs, allowing the boundary connections to be timed. Port buses to be virtualized can be specified manually in the RTL or PDC file with the port attribute, `ace_virtualize`, or automatically by ACE. Working with Virtual I/O (see page 425) contains more details.

Table 59: *Advanced Design Preparation Implementation Options (cont.)*

Option	Tcl Option	Description
Virtual IO Utilization	`virtual_io_utilization`	The I/O pad utilization percentage targeted by I/O virtualization. Must be an integer between **0** and **100**. An error is returned if the given utilization cannot be met. The value **0** requests that all possible port buses and non-bused ports are to be virtualized to achieve the smallest possible number of pads. The value **100** requests that port buses and non-bussed ports are to be virtualized until the number of remaining ports fit in the target device.
Push Flops Into Pads	`push_flops_into_pads`	Control over whether the first level of flip-flops is to be automatically pushed into the I/O pins. Automatic Flop Pushing into I/O Pads (see page 417) contains further details. **Disabled** – turns off pushing of flip-flops into the I/O pins. **Automatic** – enables full automatic pushing of all possible flip-flops into the I/O pins except for pins with the attribute "`syn_useioff=0`". **Manual** – push flip-flops only into the I/O pins which have the attribute "`syn_useioff=1`".
Pad Flop Pushing Clock Type	`pad_flop_pushing_clock_type`	Control over flop pushing into I/O pins by clock type. Automatic Flop Pushing into I/O Pads (see page 417) contains further details. **Boundary** – Only enable flop pushing into I/O pins clocked by a boundary clock. **Trunk** – Only enable flop pushing into I/O pins clocked by a trunk clock. **All** – Enable flop pushing into all I/O pins.

Table Notes

1. Setting the Period of Anti-Aging Oscillator option to a **non-zero** value increases the size of the user design in place and route and the corresponding bitstream to near maximum size.

Table 60: *Place and Route Implementation Options*

Option	Tcl Option	Description
PnR Mode	`timing_driven_pnr`	If **Timing Driven** mode is selected, data from timing analysis is used to optimize the design for high speed. If **Fast** mode is selected, placement and routing are optimized for runtime.
Multi-Threaded PnR	`mt_pnr`	Enable Multi-Threaded Place and Route. Enabling this can speed up your compile time.
PnR Seed	`seed`	The place and route seed is used to initialize the random number state in the place and route algorithms.
Placement Effort	`placement_effort`	**Low** – effort placement has a shorter runtime, but may yield less design QoR than High effort placement. **High** – effort placement increases placement runtime to further optimize the design QoR if possible.
Router Hold-Violation Fix Limit (ps)	`router_max_hold`	Specifies the maximum hold-time violation (in picoseconds) that the Router attempts to fix.
Post-PnR Buffer Limit	`max_postpnr_buffer_limit`	This limit specifies the maximum number of post-placement buffers that can be inserted.
Post-PnR Rewiring	`postpnr_rewire`	If turned on, allows post-pnr rewiring to improve the design performance and resource usage.

Table 61: *Report Generation Implementation Options*

Option	Tcl Option	Description
Output Utilization Reports	`report_utilization`	Enable utilization analysis and report generation in the flow.
Output Partition Reports	`report_partitions`	Enable partition report generation in the flow.
Output Pin Assignment Reports	`report_pins`	Enable pin assignment report generation in the flow.
Output Clock Reports	`report_clocks`	Enable clock analysis and report generation in the flow.
Output Placement Reports	`report_placement`	Enable placement report generation in the flow.
Output Routing Reports	`report_routing`	Enable routing report generation in the flow.
Output Power Reports	`report_power`	Enable power analysis and report generation in the flow.

Table 62: ***Timing Analysis Implementation Options***

Option	Tcl Option	Description
Number of critical paths	`sync_timing_num_paths`	Maximum number of critical paths per clock group.
Number of worst paths	`sync_timing_num_worst`	Maximum number of worst paths per end point.
Report unconstrained paths	`report_unconstrained_timing_paths`	When enabled, ACE includes unconstrained timing paths in the timing analysis reports.

Table 63: ***Bitstream Generation Implementation Options — Additional Outputs***

Option	Tcl Option	Description
Serial Flash (`.flash`)	`bitstream_output_flash`	Enables the generation of an additional serial flash formatted output file. This file is named the same as the STAPL file, but with a `.flash` extension. This file contains a binary image that can be directly loaded into a single serial flash memory.
4x Serial Flash (`.flash4x_[0-3]`)	`bitstream_output_4xflash`	Enables the generation of four additional 4x serial flash formatted output files. These files are named the same as the STAPL file, but with an extension ranging from `.flash4x_0` to `.flash4x_3`. Each file contains a binary image that can be directly loaded into each serial flash memory in a x4 configuration.
CPU Mode (`.cpu`)	`bitstream_output_cpu`	Enables the generation of an additional CPU Mode formatted output file. This file is named the same as the STAPL file, but with a `.cpu` extension. The file contains hexadecimal-formatted data organized in "CPU Bus Width" number of bits per file line. Data from this file is sent to the FCU CPU interface line by line (one line per clock cycle) from the top to the bottom of the file, where the left-most bit on each line is the MSB and the right-most bit is the LSB. In simulation, this file may be loaded using the `readmemh` function. For convenience, an additional binary representation of the CPU Mode output file is written, named the same as the STAPL file, but with a `_cpu.bin` extension. The file contains the same data in the same bit order as the `.cpu` file.
CPU Bus Width	`bitstream_output_cpu_width`	Controls the bit width of the CPU-mode formatted output file. When using the CPU interface in ×8 mode, set this value to **8**. If using the CPU interface in ×128 mode, set this to **128**. The value determines how many bitstream bits are printed per line in the `.cpu` output file. The bit sequence required by the FCU (and output in the generated bitstream file) may be different for each CPU Bus Width setting. Therefore, it is important to set this option to match the actual CPU hardware interface width.
Raw Hex (`.hex`)	`bitstream_output_hex`	Enables the generation of an additional Raw Hex formatted output file. This file is named the same as the STAPL file, but with a `.hex` extension. This file is used for debug purposes.

Table 64: ***Bitstream Generation Implementation Options — JTAG Scan Chain***

Option	Tcl Option	Description
Single Device Chain	`bitstream_single_device`	Specifies whether the bitstream STAPL file is output for a single-device JTAG scan chain (the target device is the only device on the JTAG scan chain). Set this to **1** to indicate a single device. If this option is set to **0** (indicating multiple devices in the scan chain), either the chain description file is used or the pre-IR, post-IR, and chain offset options are used to generate the bitstream STAPL file with knowledge of the scan chain.
Use JESD32 Chain Description File	`bitstream_use_chain_file`	When using a multi-device JTAG scan chain, specifies whether to use a JESD32 chain description file, or to use the explicit pre-IR, post-IR, and chain offset implementation options.
Chain Description File	`bitstream_chain_file`	This option specifies the optional JESD32 chain description file used by the bitstream generator to automatically pad the JTAG IR chain for multi-device chains.
Chain Offset of Target	`bitstream_chain_offset`	Specifies the offset of the target device on the JTAG scan chain for multi-device chains. Setting this to **0** selects the first device on the chain, **1** selects the second device on the chain, and so on.
IR Bits Before Target	`bitstream_preir_padding`	Specifies the total number of Instruction Register bits on the JTAG scan chain prior to the target device Instruction Register. This option is used for multi-device scan chains in order to pad the IR chain properly with 1s to put other devices in bypass mode.
IR Bits After Target	`bitstream_postir_padding`	Specifies the total number of Instruction Register bits on the JTAG scan chain after the target device Instruction Register. This option is used for multi-device scan chains in order to pad the IR chain properly with 1s to put other devices in bypass mode.

> **Note**
>
> For more details about JTAG scan chain settings and download/programming device configurations, see Configuring the JTAG Connection (see page 331).

Table 65: ***FPGA Download Implementation Options***

Option	Tcl Option	Description
JTAG Device Name [1]	`download_pod_names`	Specifies, by name, the JTAG device to be used for programming. The device naming schemes are described in the Bitstream Programming and Debug Interface User Guide (UG004).

Table Notes

1. This implementation option is stored for this implementation Flow only, and thus does not affect Bitporter pod selection for the Download View (see page 51) or Snapshot Debugger View (see page 139). The pod selection for those views is a user preference, which is managed by the Configure JTAG Connection Preference Page (see page 186), and is not a per-implementation setting. See Configuring the JTAG Connection (see page 331) for more details.

Outline View

The Outline view displays a tree of all pages in the currently active IP Configuration Editor (see page 26). Each page has its own title, and an icon to indicate the cumulative validity of all IP configuration contained on that page.

The information in the tree is dynamic, and changes as corresponding values are changed in the active IP Configuration Editor. As pages in the Editor are added or removed, entries in the tree are added or removed accordingly. As values in the Editor page change validity, the validity of the corresponding page in the Outline view tree also change.

In addition to showing the page validity, the Outline view provides an alternate method for navigating between the various dynamic pages of the IP Configuration Editors (see page 26). Selecting (clicking) an item in the Outline view causes the IP Configuration Editor to turn to the associated page.

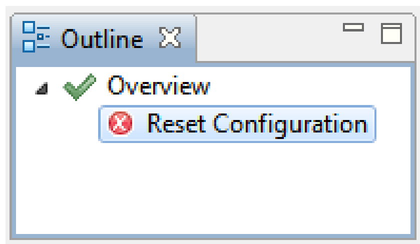


Figure 40: ***Outline View Example***

Table 66: ***Outline View Icons***

Icon	Description
	No errors or warnings on the page.
	At least one warning on the page, but no errors.
	At least one error on the page.

See also: Creating an IP Configuration (see page 306)

Package View

Warning!

The Package View is only applicable to Speedster FPGAs. This view should be ignored when developing for other Achronix product types.

The Package view provides a graphical view of the package layout of the device. This view allows visualization of the device package and placing device I/Os. The view allows zooming out to see a general overview of the package, or zooming in to see specific details.

Clicking the tall narrow arrow on the far right of the Package view shows or hides the Fly-Out Palette of display options for the Package View.

By default, the Package view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window** → **Show View** → **Other...** → **Package**.

See also: Viewing the Package Layout and Pre-Placing a Design (see page 318).

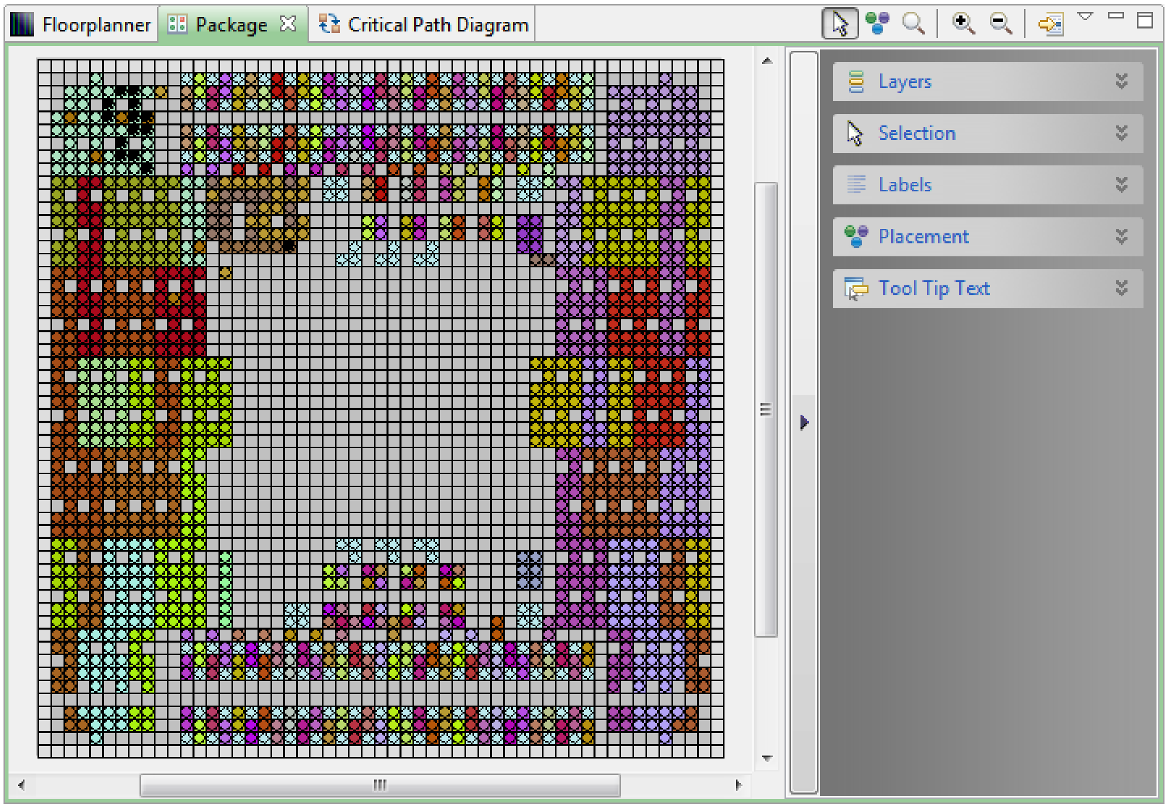


Figure 41: *Package View Example*

Table 67: *Package View Icons*

Icon	Type	Description
	Blank	No package ball at this location.
	Circle	Available ball location.
	Diamond	VDD or VSS ball location.
N/C	N/C	Non-connected ball location.

Table 68: *Package View Toolbar Buttons*

Icon	Action	Description
	Selection tool	Controls the behavior of the mouse while in the Package view. The selection tool creates a selection rectangle when the left mouse button is pressed and held. Any objects in the selection rectangle are applied with the current selection action, as configured in the fly-out palette.
	Placement tool	Controls the behavior of the mouse while in the Package view. The placement tool either pans the view or allows drag-and-drop placement of instances with the mouse when dragging while holding the left mouse button.
	Zoom tool	Controls the behavior of the mouse while in the Package view. The zoom tool creates a zoom-in rectangle with a click-and-drag action to the lower-right, and creates a zoom-out line with a click-and-drag action to the upper-left.
	Zoom in	Increases the current zoom level in the Package view by 200%.
	Zoom out	Decreases the current zoom level in the Package view by 200%.
	Save Placement	Opens the Save Placement dialog to save the current placement to a pre-placement constraints file.

Panning and Zooming

The Package view allows zooming in and out, to see more or less details respectively. There are several ways the zoom level may be changed: with the mouse scroll wheel, the ( ) **Zoom In** and ( ) **Zoom Out** buttons in the toolbar, and keyboard shortcuts are the most frequently used. See the task Zooming the Package In and Out for complete details.

Most of the other views within the Floorplanner Perspective also include context-sensitive Actions to **Zoom To** chosen individual objects or groups of objects — these actions cause the Package view to center the chosen object(s) in the view, and to change the zoom level so that the chosen object(s) are as large/detailed as possible without overflowing the visible area.

When zoomed in, the FPGA requires more area than can easily fit in the view, making it necessary to pan the view around to see the different areas of the device. Panning is most frequently performed using the arrow keys on the keyboard, or the mouse interactions with the scrollbars, or the Placement Tool drag-and-drop interactions. See the task Package Panning for complete details.

Fly-Out Palette

The following options are available in the fly-out palette in the Package view:

Layers

The ( ) **Layer** options control the layers of visible data in the Package view, allowing filtering the view to just the desired objects.

Objects in the current ACE selection set are always visible.

Table 69: *Layer Options*

Option	Default	Description
Instances	Selected	Toggles the display of all placed instances.
Toggle Groups	–	Button used to toggle the display of pin groups.
10GA	Selected	Toggles the display of the 10 Gigabit Group A I/O.
10GB	Selected	Toggles the display of the 10 Gigabit Group B I/O.
…	…	…
VSS	Selected	Toggles the display of the VSS pins.
`jtag_cfg_bank`	Selected	Toggles the display of the bank of jtag configuration pins.

The button to the right of the Layer option allows changes to the display color for that group (refer to Changing Color Coding).

Selection

The ( ) **Selection** options control the selection of objects in the Package view. The current ACE selection set is displayed in the Selection View (see page 136).

Table 70: *Selection Options*

Option	Default	Description
Select	Enabled	Controls the action applied to objects in the selection region. Causes the objects to be added to the current ACE selection set.
Deselect	Disabled	Controls the action applied to objects in the selection region. Causes the objects to be removed from the current ACE selection set.
Remove Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes the placed instances to be un-placed.

Placement

The **Placement** options control the drag-and-drop placement behavior in the Package view.

Table 71: *Placement Options*

Option	Default	Description
Fixed Placement	Enabled	Controls whether the drag-and-drop placement of an instance should be fixed or soft. Fixed placements are not changed by the placer. Soft placements are taken as a placement hint and may be changed by the placer.

> **Warning!**
>
> When pre-placing objects (for a pre-placement constraints `.pdc` file), **Fixed Placement** should always be enabled.

Tool Tip Text

The ( ) **Tool Tip Text** options control the tooltip content while hovering over objects in the Package view.

Table 72: *Tooltip Options*

Option	Default	Description
Allow Tooltips	Enabled	Allows enabling or disabling Tooltip support for the Package View without needing to toggle all the individual checkboxes.
Instance Names	Enabled	Displays the netlist instance name of placed I/O under the current mouse position in the tooltip text.
Port Names	Enabled	Displays the RTL port names of placed instances under the current mouse position in the tooltip text.
Site Names	Enabled	Displays the pad site name under the current mouse position in the tooltip text.
Ball Names	Enabled	Displays the package ball name under the current mouse position in the tooltip text.
Group Names	Enabled	Displays the I/O group name under the current mouse position in the tooltip text.
IO Port Names	Enabled	Displays the device port (FPGA I/O) name under the current mouse position in the tooltip text.
IO Use	Enabled	Indicates the group type: B = byte lane. G = general purpose. C = clock capable.
IO LVDS Polarity	Enabled	Displays the polarity (P or N) of the I/O buffer under the current mouse position in the tooltip text: P = positive polarity. N = negative polarity.
IO Clock and Reset Capability	Enabled	Displays the clock and reset capability of the I/O buffer under the current mouse position in the tooltip text: D = data only. RD = reset and data capable. CRD = clock, reset, and data capable.

Labels

The ( ) **Label** options control the text labels on objects in the Package view.

Table 73: *Label Options*

Option	Default	Description
None	Disabled	Disables the display of label text in the package graphic.
Instance Names	Disabled	Displays the netlist instance name in the package graphic.
Port Names	Disabled	This option displays the RTL port names on placed instances.
Site Names	Disabled	Displays the pad site name in the package graphic.
Ball Names	Enabled	Displays the package ball name in the package graphic.
Group Names	Disabled	Displays the I/O group name in the package graphic.
IO Port Names	Disabled	Displays the device port (FPGA I/O) name in the package graphic.
IO Use	Disabled	Indicates the group type: B = byte lane. G = general purpose. C = clock capable.
IO LVDS Polarity	Disabled	Displays the polarity (P or N) of the I/O buffer in the package graphic: P = positive polarity. N = negative polarity.
IO Clock and Reset Capability	Disabled	Displays the clock and reset capability of the I/O buffer in the package graphic: D = data only. RD = reset and data capable. CRD = clock, reset, and data capable.

Partitions View

Note

The Partitions View is only relevant when Incremental Compilation is enabled and partitions (compile points) have been defined in the project constraints files. Otherwise, this view table is empty.

The Partitions View displays the state of the active implementation partitions, and allows (through interactions with the Floorplanner View (see page 53), Search View (see page 132), and Selection View (see page 136)) visualizations of the partitions and their relationships with the rest of the active implementation.

The Partitions View is a default member of the Floorplanner perspective, and can be added to any other perspective by selecting **Window** → **Show View** → **Other...** → **Partitions View**.

See also: Using Incremental Compilation (Partitions) (see page 363)

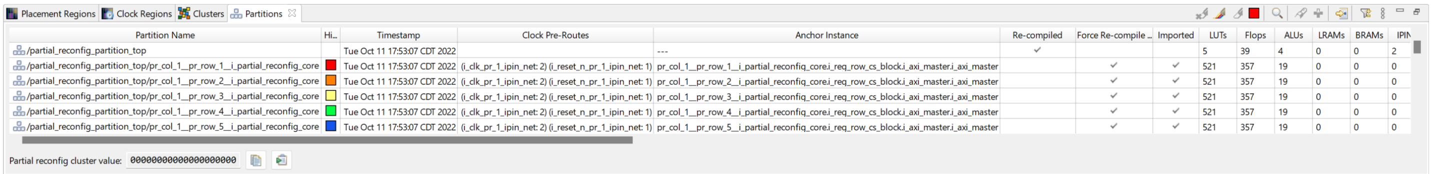


Figure 42: *Partitions View Example*

> **Caution!**
>
> Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The screenshots and example descriptions in this section do not reflect the resource types of the actual device being used.

Table 74: *Partitions View Columns*

Column	Description
Partition Name	The name of the partition as specified in the design constraints file(s).
Highlight Color	If all instances within the partition have the same highlight color, the row shows a color square with that same highlight color. If even one contained instance has a differing highlight color, or no highlight at all, then the row displays no color square.
Time Stamp	The timestamp of the last compile for this partition (compile point) in the upstream synthesis tool.
Clock Pre-Routes	If any clock pre-routes exist for a given partition, they are listed here.
Anchor Instance	The instance in the partition to "anchor" drag-and-drop operations.
Re-compiled	Contains a checkmark if the partition was recompiled, requiring placement and routing to be re-run.
Force Re-compile on Next Run	Indicates whether the partition is forced to re-compile during the next pass through ACE. This checkbox may be toggled on or off using the right-click Context Menu choices **Force Partition Changed** and **Un-Force Partition Changed**.
Resource	The sum count of all *resource* instances contained in this partition and no other partitions.
Cumulative *Resource*	The sum count of all contained *resource* instances, including in child partitions (below this partition in the RTL hierarchy).

A number of actions are available within the view, available in the toolbar at the top of the view and/or in right-click context menus for each partition in the table.

Table 75: *Partitions View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	Search for Instances	Y	Y	Issues a `find` Tcl command that returns the names of all the instances in the selected partition(s).
	Add Instances to Selection	Y	Y	Adds the instances within the partition to the ACE Selection Set (as shown in the Selection View (see page 136)).
	Highlight	Y	Y	Applies the currently active Highlight color to the instances within the chosen partition (see Highlighting Objects in the Floorplanner View (see page 315)).
	Un-Highlight	Y	Y	Clears the Highlight for the instances within the chosen partition.
	Choose Highlight Color	Y	Y	Determines which color is applied to the objects chosen the next time the Highlight action is selected for this view.
	Auto-Highlight	Y		Automatically assigns a unique highlight color to each partition.
	Zoom To	Y	Y	Zooms the Floorplanner view to a region containing the instances within the partition currently chosen in the tree.
	Toggle Filter Row Visibility (1)	Y		Changes whether the filter row (of filter icons) is visible.
	Configure Clock Pre-Routes		Y	Allows adding clock pre-route information for the selected partition(s).
	Force Partition Changed		Y	Override the partition timestamp during the next pass through Ace: A check mark appears in the **Force Re-compile on Next Run** column, and the partition is re-placed and re-routed the next time the flow is run, even if there were no RTL changes and it was not re-compiled in the upstream synthesis tool. **Un-Force Partition Changed** removes the check mark in the column.
	Un-Force Partition Changed		Y	Removes the check mark from the **Force Re-compile on Next Run** column, so the compilation timestamp is no longer overridden. This is essentially an undo operation for the **Force Partition Changed** action.
	Export Partition		Y	Exports the selected partition to an `.epdb` file under `{impl_name}/output/partitions`, and exports a blackbox netlist for the partition to a `.v` file under `{impl_name}/output/blackboxes`.

Table Notes

1. Toggle Filter Row Visibility does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Note

All actions upon a partition act only upon the members of that partition, not upon the members of any child partitions.

Drag-and-Drop

The Partitions view supports a limited set of Drag-and-Drop interactions with other views in the Floorplanner perspective (see page 24). The view only acts as a Drag-and-Drop source; items dropped on the Partitions view are ignored.

Any row of the table may be dragged to the Tcl Console view (see page 144), and when dropped anywhere in the view the partition name (with the appropriate object type prefix) is inserted at the beginning of the Tcl command-line.

Any partition in the table may be dragged to the Placement Regions view (see page 120) or the Floorplanner View (see page 53) (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 358). Dropping in the Placement Regions view uses the `add_region_find_insts` Tcl command. Dropping onto the Floorplanner view uses the `set_placement -partition` Tcl command. Dragging a partition is the equivalent of dragging all individual instances which are members of that partition.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all selected partition rows in the view. Selecting more or fewer rows in the view updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Placement Regions View

The Placement Regions view provides a tabular representation of the content of all user-created Placement Regions (see page 354) for the design. The view allows the manipulation of the visibility of the Placement Region itself as painted (as a colored overlay) in the Floorplanner View (see page 53), and the content (the instances constrained to the region) of each Placement Region. The view table remains empty until the currently active Implementation has completed the **Run Prepare** flow step.

Because Placement Regions are manually defined, and Instances are manually constrained to the Placement Regions, it is necessary to track the total number of sites and associated Instances for each Resource type. Accordingly, based upon the chosen Target Device Implementation Option, there are columns for each available Resource type found within the device. If more Instances are ever assigned to a Placement Region than there are available sites of that type within that Placement Region, the view displays an error for that placement region and resource type combination.

The section Placement Regions and Placement Region Constraints (see page 354) describes the creation and usage of Placement Regions in greater detail.

By default, the Placement Regions view is included in the Floorplanner perspective (see page 24). To add the view to another perspective, select **Window → Show View → Other... → Achronix → Placement Regions**.

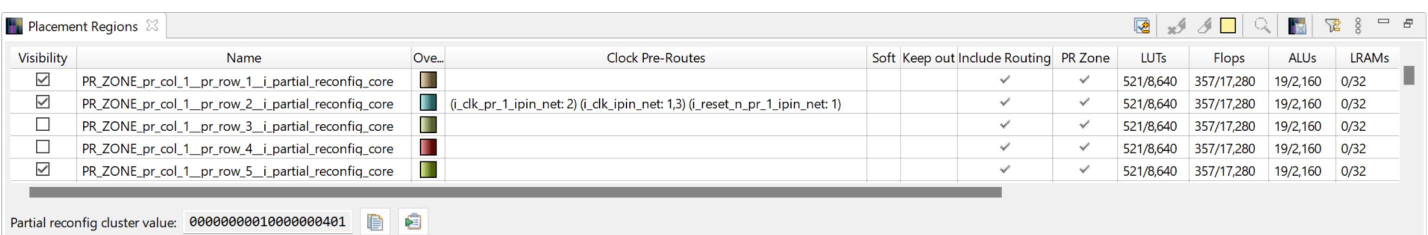


Figure 43: *Placement Regions View Example*

In the above example screenshot, error icons are shown for the `CLK_IPIN` assignments of `region_2`, indicating that too many instances (1) are assigned for the available sites (0) within the region. The region itself also shows an error icon to indicate when an erroneous resource type is scrolled offscreen.

> **Note**
>
> Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic. When the Run Prepare flow step has completed, the columns appropriate to the target device are chosen and values are updated. The resource type columns shown in the screenshots should be considered examples only — they might not match the exact resources of any particular target device.

Table 76: ***Placement Regions Table Columns***

Column	Description
Visibility	When selected (this is user-editable), this placement region overlay is painted in the Floorplanner View (see page 53), using the chosen translucent **Overlay Color**.
Name	The name of this placement region.
Overlay Color	The (user editable) translucent color used to paint an overlay in the Floorplanner View, showing the location of the placement region.
Clock Pre-Routes	If any clock pre-routes exist for a given partition, they are listed here.
Soft	When the placement region is created, it can be defined as a Soft region. Soft regions contain a checkmark in this column. The "soft" placement region status cannot be changed after creation.
Keep Out	When the placement region is created, it can be defined as a Keep Out region. Keep Out regions contain a checkmark in this column. The "keep out" placement region status cannot be changed after creation.
Include Routing	Treats the region as a routing constraint as well as a placement constraint, keeping all routing wires and instances inside the region boundary box.
PR Zone	When the placement region is created, it can be defined as a Partial Reconfiguration (PR) zone. PR Zone regions contain a checkmark in this column. The "PR zone" placement region status cannot be changed after creation.
Resource	The number of *Resource* Instances constrained to this placement region or the number of *Resource* sites contained within the bounds of this placement region.

Table 77: ***Placement Regions View Actions***

Icon	Action	Toolbar Button	Context Menu	Description
	Show/Hide overlay: `region_name`		Y	Toggles the **Visibility** checkbox for this placement region, showing or hiding the colored translucent overlay for the placement region in the Floorplanner View (see page 53).
	Show or Hide All	Y		Toggles the **Visibility** checkbox for all placement regions, showing or hiding their colored translucent overlays in the Floorplanner View.
	Select Constrained Instances		Y	Adds all Instances constrained within this placement region to the ACE Selection Set (see Selection View (see page 136)).

Table 78: *Placement Regions View Actions (cont.)*

Icon	Action	Toolbar Button	Context Menu	Description
	Deselect Constrained Instances		Y	Removes all Instances constrained within this placement region from the ACE Selection Set.
	Change Overlay Color		Y	Allows the choice of which translucent Overlay Color is used to represent this placement region in the Floorplanner View.
	Reset Overlay Color		Y	Resets the chosen placement region overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two placement regions are too similar for easy discernment, another color is pseudo-randomly picked. Each time this action is chosen, another color is picked.
	Reset All Overlay Colors		Y	Pseudo-randomly reassigns new overlay colors for all placement regions.
	Un-Highlight Constrained Instances	Y	Y	Clears the opaque Highlight color for all instances constrained to the selected Placement Region.
	Highlight Constrained Instances	Y	Y	Highlights all instances constrained to the currently selected placement region with the currently-selected opaque highlight color. The highlighted results are visible in the Floorplanner view (see Highlighting Objects in the Floorplanner View (see page 315)).
	Choose Highlight Color for next Highlight command	Y	Y	Allows the changing of the current placement region constrained instances opaque highlight color (which is different from the placement region translucent overlay color). This opaque highlight color is used in the Floorplanner view when the **Highlight Constrained Instances** ( ) action is chosen in the Placement Regions view.
	Zoom to: `region_name`	Y	Y	Zooms and pans the Floorplanner view to show the location of the selected Placement Region (the Placement Region itself is not visible as an overlay unless the appropriate **Visibility** checkbox is enabled in the Placement Region View table).
	Print Instances: `region_name`		Y	Causes a list of all Instances constrained to the selected placement region to be printed in the Tcl Console. Issues the `get_region_insts` Tcl command.
	Remove All Instance Constraints		Y	Removes all Instances from this Placement Region, thus clearing their placement region constraints.
	Delete Placement Region		Y	Removes the selected Placement Region and all associated placement region constraints from the design.
	Configure Clock Pre-Routes...		Y	Allows adding clock pre-route information to the selected partition(s).
	Save Placement Regions	Y		Brings up the Save Placement Regions Dialog (see page 172) to save one or all of the Placement Regions definitions and all associated instance placement region constraints.
	Toggle Filter Row Visibility	Y		Changes whether the filter row (of filter icons) at the top of the table is visible or not. This does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Using the Table to Display Placement Regions in the Floorplanner View

Each Placement Region is automatically given a unique translucent overlay color to represent the Placement Region when painting the Floorplanner View (see page 53). By default, no Placement Region overlays are painted in the Floorplanner View. The Placement Region overlays to be displayed must be enabled. The overlay color may optionally be altered for each or all Placement Regions, but these color choices are not persistent between ACE sessions.

> **Note**
>
> While alternate overlay colors may be chosen for each Placement Region, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each Placement Region.

The following are ways to alter the presentation of Placement Region data in the Floorplanner View (see page 53):

Enable/Disable Painting of Individual Placement Regions Within the Target Device

When the checkbox in the **Visibility** column for a Placement Region is selected, the area of the target device (in the Floorplanner view) representing that Placement Region is painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view is redrawn with the chosen Placement Region overlay no longer painted.

Enable/Disable Painting of all Placement Regions Within the Target Device

When the ( ) **Show/Hide All Placement Regions** action is chosen, the visibility of all Placement Regions is simultaneously either enabled or disabled, causing the Floorplanner View to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Placement Regions

The overlay rendering color of each individual Placement Region may be chosen as follows:

1. Right-click anywhere on the row of the desired Placement Region.
2. Select **Choose Overlay Color** from the popup context menu.
3. Use the Color Dialog to choose the desired color for the Placement Region.

This is a temporary color change — colors are reverted to automatically chosen defaults every time ACE starts. During a session, colors are also reverted to the defaults if any of the following are changed:

- The active design
- The active implementation
- The target device

ACE automatically picks a different overlay color for an individual Placement Region if **Reset Overlay Color** is chosen from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Placement Regions

ACE automatically picks different overlay colors for all Placement Regions if the **Reset All Overlay Colors** action is chosen.

Organizing Table Data

The following are ways to alter the presentation of the data in the Placement Regions table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns. The mouse cursor changes to indicate resizing is possible.
2. Click and drag left or right to resize the column to the desired width.
3. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Click and hold any column name.
2. Drag left or right to move the column between any other pair of columns.
3. Release the mouse button to insert the column header at the new location.

While dragging, the dragged column header appears alongside the mouse cursor. A thick column header separator appears at the present cursor location to show where the column insertion occurs if the mouse button is released.

Filtering

Most table columns can filter the displayed Placement Regions by value. When filtering by column value, only Placement Regions with column values matching the filter are retained; non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the Overlay Color column) lack a filter icon in the filter row.

To add a filter to a column:

1. The Filter Row must first be visible. Select the ( ) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the ( ) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the ( ) active filter icon.

To edit (or clear) an existing filter:

1. Click the ( ) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.

4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the ( ) inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all placement regions marked as "visible" in the view. Making more or fewer placement regions visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Projects View

The Projects view provides a hierarchical view of the Projects (see page 215) in the Workbench (see page 24). From here, projects can be added and removed, project configurations edited, the active implementation (see page 215) can be chosen, saved, or restored, files may be opened for editing, etc.

Clicking an implementation activates (see page 221) it. Similarly, clicking a project activates the first implementation in the project definition. The active project and active implementation are both displayed in a bold font.

The various Source Files (see page 217) making up a project are added and removed from this view. Source files of the project are listed in the order in which they are loaded. To change the order the source files are listed or loaded, see Adding Source Files (see page 272).

By default, the Projects view is included in the Projects perspective (see page 24). To add it to the current perspective, click **Window** → **Show View...** → **Projects**.

For detailed information about managing projects, implementations, source files, etc., see Working with Projects and Implementations (see page 268). See also: Project Management Preference Page (see page 210).

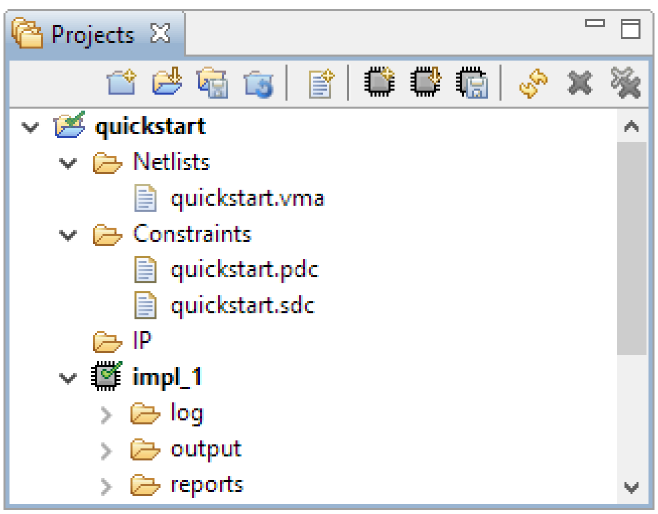


Figure 44: ***Projects View Example***

Table 79: *Projects View Actions*

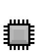
Icon	Action	Toolbar Button	Context Menu	Description
	Open File		Y	Opens the selected file in a text editor within ACE. See also: display_file (see page 545).
	Create project	Y		Opens the Create Project Dialog (see page 162) to allow creating a new project definition in the tool. See also: create_project (see page 543).
	Load project	Y		Opens the Load Project Dialog (see page 165) to allow loading an existing ACE Project File into the tool. See also: load_project (see page 570).
	Save project	Y	Y	Saves the changes to the selected ACE Project to its ACE Project File on disk. See also: save_project (see page 599).
	Save Project As...		Y	Saves the selected ACE project to a newly-chosen filename and location on disk.
	Reload project	Y	Y	Reloads the selected ACE Project. See also: restore_project (see page 584).
	Add source files [(1)]	Y	Y	Opens the Add Source Files Dialog (see page 147) to allow adding source netlist and constraint files to the selected project in the Projects view. This action also allows adding IP Configuration (see page 306) files (`.acxip`) to the project as a convenience. See also: add_project_ip (see page 535), add_project_netlist (see page 535) and add_project_constraints (see page 534).
	Create implementation	Y	Y	Opens the Create Implementation Dialog (see page 158) to allow creating a new project implementation definition for the selected project in the Projects view. See also: `create_impl` Tcl command.
	Restore implementation	Y	Y	Opens the Restore Implementation Dialog (see page 167) to allow restoring the active project implementation from an Acxdb Archive File. See also: `restore_impl` Tcl command.
	Rename implementation		Y	Allows renaming the Implementation. See also: `rename_impl` Tcl command.
	Save implementation	Y	Y	Opens the Save Implementation Dialog (see page 169) to allow saving the active project implementation to an Acxdb Archive File. See also: `save_impl` Tcl command.
	Open Multiprocess Report		Y	Opens the Multiprocess Summary Report (see page 238) for the selected project, if the project has one. See also: `display_file` Tcl command.
	Refresh contents	Y	Y	Refreshes the listing of supporting files contained within the selected project or implementation.

Icon	Action	Toolbar Button	Context Menu	Description
	Remove	Y	Y	Allows removing the selected items from the Projects view. Removing items from a project does not delete the corresponding resources from the file system, except for removing implementation output and report files. See also: `remove_project`, `remove_project_constraints`, `remove_project_netlist`, `remove_project_ip` and `remove_impl` Tcl commands.
	Clone IP		Y	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP		Y	Renames the selected IP.
	Add IP to another project...		Y	Adds the selected IP to another project in the ACE workspace.
	Add copies of IP to another project...		Y	Adds a copy of the selected IP to another project in the ACE workspace.
	Regenerate All IP Design Files		Y	Regenerates HDL (Verilog and optionally VHDL), constraint files (`*.pdc` and `*.sdc`), etc. for all IP Design files contained in the project (as found in the IP folder of the project in the Projects view). See also: `generate_ip_design_files` Tcl command.

Table Notes

1. Constraint files are loaded by ACE in the same order they are displayed within this view. For details on how to change this display/load order, see Adding Source Files (see page 272).

Table 80: *Project View Icons*

Icon	Description
	Project
	Project (Active)
	Project (Save Needed)
	Project (Active, Save Needed)
	Implementation

Icon	Description
	Implementation (Active)
	File

Note

Some files in the "Constraints" section of the tree may appear greyed-out to indicate that those constraint files are not enabled in the Active Implementation (see page 221) as shown in the following figure. Various constraint files in a project can be enabled or disabled for an implementation in the Options View (see page 103), under **Design Preparation → Constraint Files**.

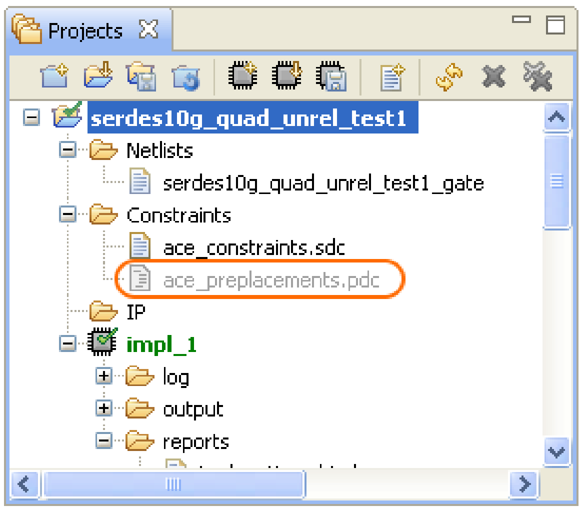


Figure 45: ***Disabled Constraints File Example***

Properties View

The Properties View can provide in-depth specifics about many types of pin, net, and instance items on demand, and the view then allows navigating many of the relationships between connected items.

To initialize the Properties View with desired information, use the **Display Properties For...** choices found on the right-click context menus of many of the views within the Floorplanner Perspective. The Tcl command `display_properties` can also be used to populate the Properties View.

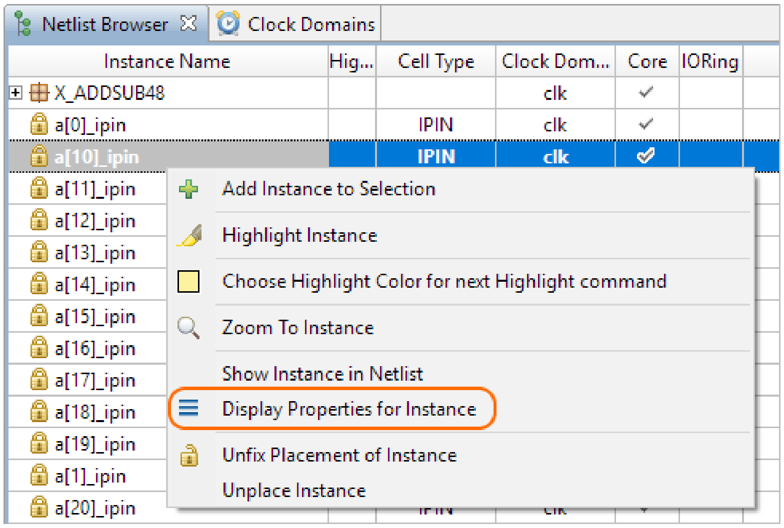


Figure 46: ***Properties View Example***

General Tab

The **General** tab shows the basic, top-level information about the item.

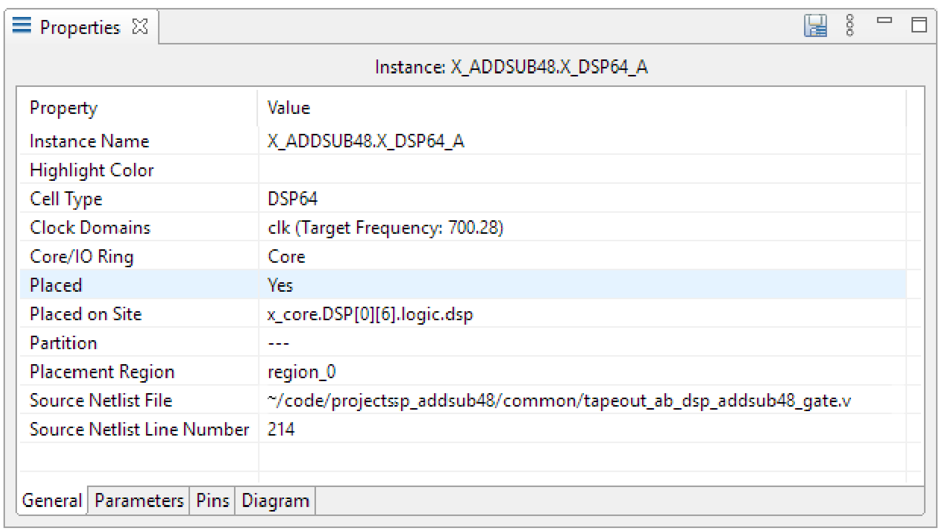


Figure 47: ***General Tab Example***

> **Note**
>
> Double-click a **Source Netlist File** filename to immediately open that file in the Editor area, or double-click a **Source Netlist Line Number** to immediately open the source netlist file, showing the given line number, in the Editor area.

Parameters Tab

The **Parameters** tab shows the type, name, and value of all of the configurable parameters for the item.

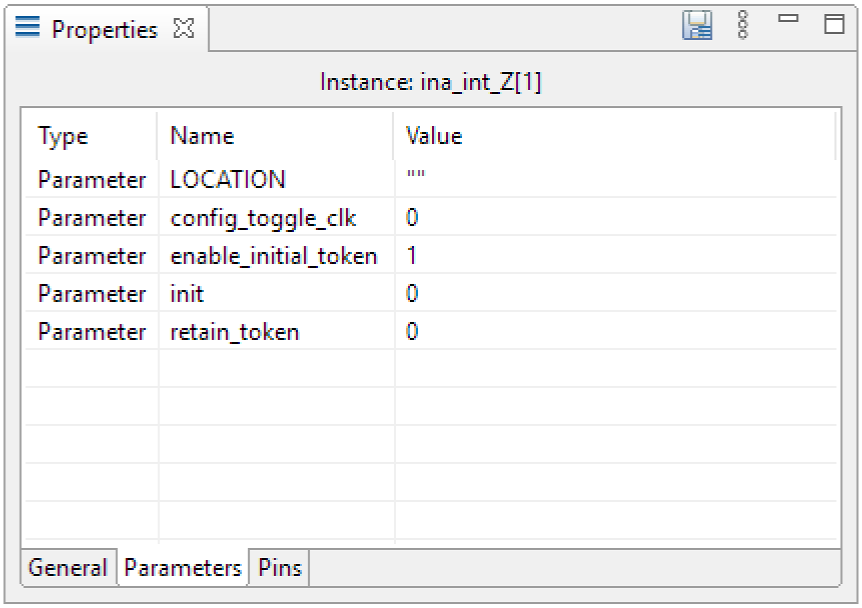


Figure 48: ***Parameters Tab Example***

Pins Tab

The **Pins** tab shows a variety of information about the item pins.

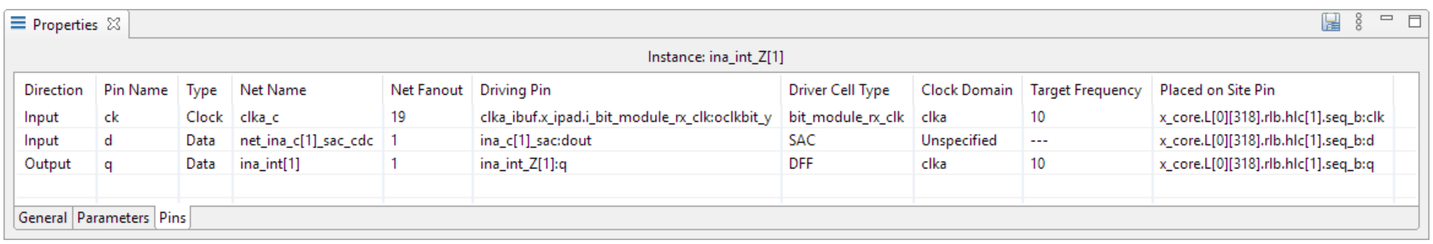


Figure 49: ***Pins Tab Example***

A number of actions are available on the Pins tab right-click menus.

Table 81: ***Properties View Pins Tab Actions***

Icon	Action	Advanced	Description
	Copy Cell Text		Copies the text onto the system clipboard.
	Add to Selection		Adds the item to the ACE selection set (as shown in the Selection View (see page 136)).
	Remove from Selection		Removes the item from the ACE selection set (as shown in the Selection View (see page 136)).
	Highlight		Applies the currently active highlight color to the chosen item (see Highlighting Objects in the Floorplanner View (see page 315)).

Icon	Action	Advanced	Description
	Choose Highlight Color		Determines which color is applied the next time the Highlight action is selected.
	Zoom To		Zooms the Floorplanner view to a region containing the item.
	Show in Netlist [(1)]		Attempts to open a text editor to the file and line number relevant to the chosen item (available only when a single item is chosen in the view).
	Display Properties		Display properties for the chosen item in the Properties view.

Table Notes

1. **Caution:** The Show in Netlist feature is Early Access functionality and might not always open the text editor to the expected location.

Note

Reminder: Instances Selection color vs Highlight color priority

- With default preference settings, in the Floorplanner View (see page 53), highlight colors of (placed) instances are only visible when the Instances Layer is enabled, and the instances are not members of the ACE selection set. This behavior is due to the instance selection color having a higher priority than the highlight color.

Properties Navigation

- Move from one item to another by using the **Display Properties** right-click menu items in the Pins tab.
- Use **Display Properties** to move between related pins, nets, and instances.

Diagram Tab

Some items are complicated or interesting enough to warrant a supplemental diagram. For these types of items, a diagram showing the current item configuration can be found on the **Diagram** tab. Tooltips over the diagram can provide supplemental information where useful.

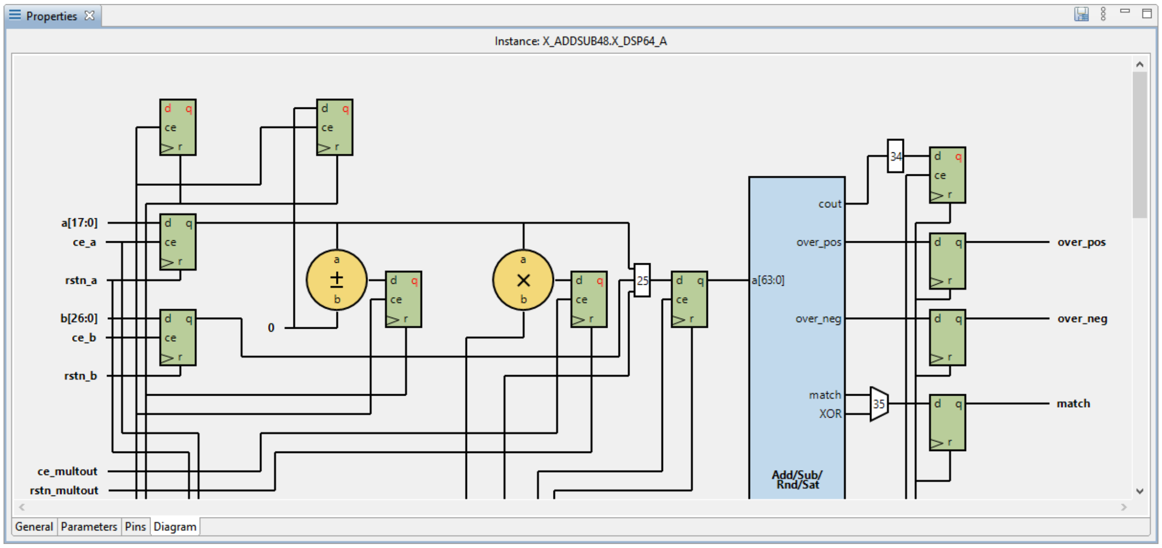


Figure 50: *Diagram Tab Example*

Save Properties

The **Save Properties** action can be used to save all changed properties on objects in the database after prepare has been run. See the `save_properties` Tcl command reference for details.

This action can be performed by clicking **Save Properties** on the Properties view tool bar:

Figure 51: *Save Properties Example*

Search View

The Search view provides an interface for searching the ACE design database for design objects (instances, nets, ports, pins, sites, and paths), displaying the results of a search in a list, organized by object type. Optionally, all or part of the results of a search can be added to the current ACE Selection Set, as displayed in the Selection view (see page 136). The Search View is a graphical interface to the `find` Tcl command.

Instances and Ports in the results list may be dragged and dropped onto the Floorplanner View (see page 53) to assign placement or add placement region (see page 354) constraints (the behavior depends upon the Floorplanner active tool /mode). Instances and Paths in the results list may be dragged to the Placement Regions View (see page 120) to add placement region constraints.

By default, the Search view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window** → **Show View...** → **Search**.

See also: Object Type Prefixes (see page 305)

> **Note**
>
> A maximum of 200 objects are displayed in the Search view at a time. Use the arrow buttons (⇦ and ⇨) on the view toolbar to page through the full set of search results.

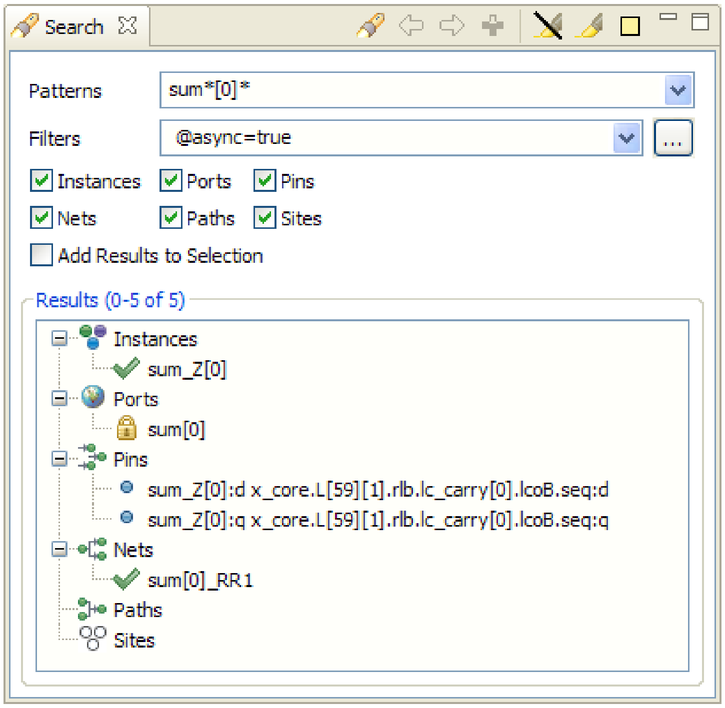


Figure 52: ***Search View Example***

Table 82: *Search View Icons*

Icon	Description
	Object (unplaced instances and ports; all pins, nets, and paths).
	Placed Object (applies to instances and ports).
	Fixed-Placed Object (applies to instances and ports).
	I/O Macro (applies to ports).
	Instances (all instances are under this branch of the search results).
	Ports (all ports are under this branch of the search results).
	Pins (all pins are under this branch of the search results).
	Nets (all nets are under this branch of the search results).
	Paths (all paths are under this branch of the search results).
	Sites (All sites are under this branch of the search results).

Many of the actions in the Selection view are available as both toolbar buttons and right-click context menu choices. Toolbar buttons typically act upon all the listed Search results items, while context menu actions only affect the subset of items currently chosen within the Results list.

Table 83: *Search View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	Find objects	Y		Searches for objects in the ACE design database using the search criteria from the Search view.
	Display next 200 results	Y		Displays the next 200 objects in the search results list.
	Display previous 200 results	Y		Displays the previous 200 objects in the search results list.
	Add to selection	Y	Y	Adds all objects that are currently chosen in the Search view "Results" list to the current selection set (as displayed in the Selection View).
	Remove from selection		Y	Removes all objects that are currently chosen in the Search view "Results" list from the current selection set (as displayed in the Selection View).
	Un-highlight Results (1)	Y	Y	Turns off the highlight color for objects.
	Highlight Results	Y	Y	Highlights objects with the currently-selected search highlight color. The highlighted results are visible in the Floorplanner View (other views are not affected by highlights).
	Choose Highlight Color	Y		Allows changing the current highlight color for search result highlighting. This color is used in the Floorplanner View when the Search view ( ) **Highlight Results** button is clicked.
	Zoom To Object		Y	Zooms the Floorplanner view to a region containing the item currently chosen in the results list.
	Show in Netlist (2)		Y	If relevant data exists, opens a text editor to the file and line number relevant to the chosen result item (available only when a single item is chosen in the results list, and that item is an Instance or Net).
	Fix Placement of Instance		Y	Causes the placement state of the chosen Instance to change from unfixed (or soft) to Fixed.
	Unfix Placement of Instance		Y	Causes the placement state of the chosen Instance to change from Fixed to unfixed (soft).

Icon	Action	Toolbar Button	Context Menu	Description
	Unplace Instance(s)		Y	Unplaces all Instances currently chosen in the results list.
	Save Placement Using Search Results	Y		Opens the Save Placement Dialog (see page 170), pre-populating its **Specific List of Instances** field with the current search query.

Table Notes

1. The Un-highlight Results button stops highlighting the search results in the Floorplanner View. Other views are not affected by highlighting.
2. The Show in Netlist action is Early Access functionality and might not always open the text editor to the expected location.

Table 84: ***Search View Options***

Option	Description
Patterns	Enter a Tcl regular-expression pattern which is used to perform a name-based search. Previously used search patterns may be selected from the drop-down menu.
Filters	Enter a search filter to further restrict the search results by properties other than name. Previously used search filters may be selected from the drop-down menu. See Filter Properties (see page 244).
... (Search Filter Builder)	This button opens the Search Filter Builder Dialog (see page 174) providing a guide through the options available for search filters.
Instances (1)	When checked, includes Instances in the search results.
Ports (1)	When checked, includes Ports in the search results.
Pins (1)	When checked, includes Pins in the search results.
Nets (1)	When checked, includes Nets in the search results.
Paths (1)	When checked, includes Paths in the search results.
Sites (1)	When checked, includes Sites in the search results.
Add Results to Selection	If selected when a search is performed, all the results of that search are added to the ACE selection set.

Option	Description

Table Notes

1. **Caution:** If none of the object-type option checkboxes are checked, the search is performed as if all types were checked.

Search Results and ACE Selection

The complete results of a search may be added to the current ACE Selection Set (and thus rendered in a special color, by default a bright green, in the Floorplanner View) by checking the **Add Results to Selection** checkbox before starting the search. A subset of the search results may be added to the current ACE selection set by selecting the desired additions in the search "Results" list and pressing the (✚) **Add to Selection** button, or, double-clicking a single entry in the "Results" list adds it to the current selection.

Search Highlights

There is typically a tremendous amount of visualization data available in the Floorplanner view. The granular highlighting allowed by the Search view, the Selection view, and the Highlight functionality (see Highlighting Objects in the Floorplanner View (see page 315)) is an attempt to help turn this data into useful information, to find and focus on specific information within the user designs.

By highlighting multiple search result sets in the same or different colors, desired information is made more visible in the graphical views. By selectively un-highlighting or re-highlighting smaller (more specific) result sets (which are a subset of already-highlighted objects), focus may be directed to just the objects of interest.

When used in combination with the layering functionality (see the Layers portion of the toolbox for the Floorplanner view) and Selection functionality (see the Selection view (see page 136) as well as the `select` and `deselect` Tcl commands), a graphical visualization can be achieved at whatever granularity is desired.

Caution!

The Selection color takes precedence over the Highlight color by default. If design objects are both highlighted and selected, they are painted the selection color (bright green by default) in the Floorplanner view. To see the design objects painted in the Highlight color (with default precedence settings), the objects must first be removed from the current Selection set (as shown in the Selection view). The Floorplanner view settings (including precedence) for Highlight and Selection colors can be manipulated on the Floorplanner View Colors and Layers Preference Page (see page 190).

Selection View

The Selection view provides an interface that allows viewing and managing the current selection set. A selection set consists of a collection of ACE design database objects. The selection set may also be manipulated with the `select` and `deselect` Tcl commands.

The Selection view displays the current selection set in a list, organized by object type. The object type groupings are Instances, Ports, Pins, Nets, Paths, and Sites; these are the only object types which may be Selected.

> **Note**
>
> A maximum of 200 objects are displayed in the Selection view at a time. Use the arrow buttons (⇦ and ⇨) on the view toolbar to page through the full content of the ACE selection set.

The (current page of) selected objects in the Selection view is also displayed with special coloration (by default a bright green) in the Floorplanner view (see page 53).

Objects may be added to the selection set from the Search view (see page 132) (if **Add Results to Selection** is checked when a search is issued, or by choosing individual objects from the search results and selecting **Add to Selection**), or from the Floorplanner view (see Selecting Floorplanner Objects (see page 313)).

Various drag-and-drop operations may be initiated by dragging single or (in some cases) multiple items from the selection list to other views in the Floorplanner perspective. If dragging all selected objects of a given type (i.e., Instances), including those not in the current page of 200 selected objects, the node with that type name (i.e., Instances) may be dragged. (Be aware that some drag-and-drop operations, such as pre-placement assignment, does not work with multiple, simultaneously selected objects.)

Instances and Ports in the selection list may be dragged and dropped onto the Floorplanner view to assign pre-placement or add placement region (see page 354) constraints (the behavior depends upon the active Floorplanner tool or mode). Instances and Paths in the results list may be dragged to the Placement Regions view (see page 120) to add placement region constraints.

By default, the Selection view is included in the Floorplanner perspective (see page 24). To add it to the current perspective, select **Window** → **Show View...** → **Other...** → **Selection**.

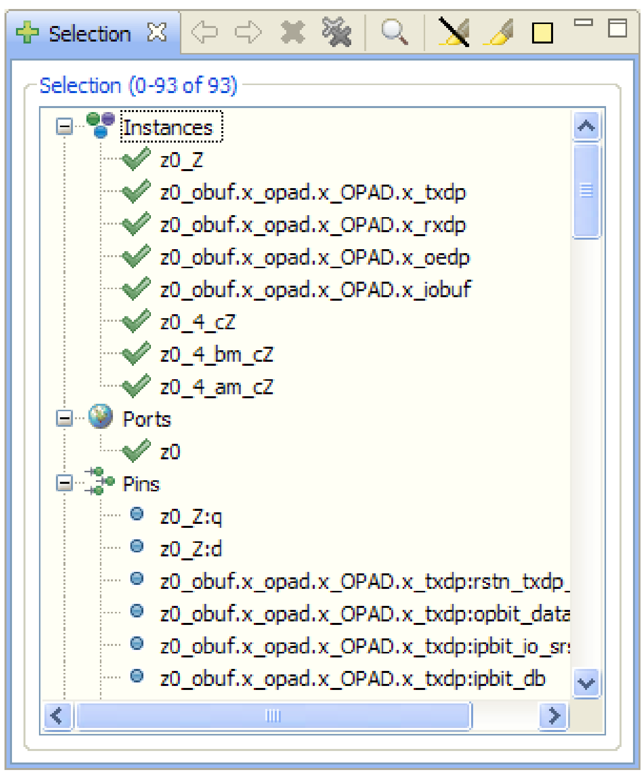


Figure 53: *Selection View Example*

Table 85: *Selection View Icons*

Icon	Description
	Object.
	Placed Object (applies to instances and ports).
	Fixed-Placed Object (applies to instances and ports).
	I/O Macro (applies to ports).
	Instances (all instances are under this branch of the selection).
	Ports (all ports are under this branch of the selection).
	Pins (all pins are under this branch of the selection).
	Nets (all nets are under this branch of the selection).
	Paths (all paths are under this branch of the selection).
	Sites (all sites are under this branch of the selection).

Many of the actions available in the Selection view are available as both toolbar buttons and right-click context menu choices. Toolbar buttons act upon all the listed Selection items, while context menu actions only affect the subset of items currently chosen within the Selection list. Be aware that available right-click context menu choices vary depending upon the context: the number and the type of the items alter the available actions.

Table 86: *Selection View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	Zoom to Full Selection Set	Y		Zooms the Floorplanner view to a region containing the current list of chosen objects in the Selection view.
	Zoom to Object		Y	Zooms the Floorplanner view to a region containing the currently chosen object in the Selection view.
	Display next 200 objects	Y		Displays the next 200 objects in the selection set.
	Display Previous 200 objects	Y		Displays the previous 200 objects in the selection set.
	Deselect object	Y	Y	Deselects objects in the Selection view list, removing them from the current selection set in ACE.

Icon	Action	Toolbar Button	Context Menu	Description
	Deselect all objects	Y		Deselects all objects in the current selection set in ACE, resulting in an empty selection set.
	Un-Highlight Selection (1)	Y	Y	Turns off the highlight color for objects in the current selection.
	Highlight Selection	Y	Y	Sets the highlight color for objects in the current ACE selection set to the currently-chosen highlight color (the highlight coloring is only visible in the Floorplanner view after the objects are no longer Selected, since the Selection color overrides the highlight color.
	Choose Highlight Color	Y		Allows changing the current ACE selection set highlight color (which is different from and overridden by the Selection color). This color is used in the Floorplanner view when the ( ) **Highlight Selection** action is chosen in the Selection view.
	Show in Netlist (2)		Y	If relevant data exists, opens a text editor to the file and line number relevant to the chosen Selection item (available only when a single item is chosen in the Selection list, and that item is an Instance or Net).
	Fix Instance Placement		Y	Causes the placement state of the Instance under the mouse cursor to change from unfixed (or soft) to Fixed.
	Unfix Instance Placement		Y	Causes the placement state of the Instance under the mouse cursor to change from Fixed to unfixed (or soft).
	Unplace Instance(s)		Y	Unplaces the Instances chosen in the view.
	Unplace All Instances in ACE Selection Set		Y	Unplaces all Instances that are members of the current ACE selection set.
	Save Placement of Selection Set	Y		Opens the Save Placement Dialog (see page 170), pre-populating its **Specific List of Instances** field with the query to obtain the active Selection Set.

Table Notes

1. Stops highlighting the ACE selection set in the Floorplanner view. Other views are not affected by highlighting.
2. The Show in Netlist action is Early Access functionality and may not always open the text editor to the expected location.

For more information about the interaction between Selection and Highlighting, please see Search Highlights (see page 136) as well as Highlighting Objects in the Floorplanner View (see page 315).

See also: Object Type Prefixes (see page 305).

Snapshot Debugger View

The Snapshot Debugger view provides a graphical interface for controlling an embedded Snapshot IP block in a programmed Achronix device. By default, the Snapshot Debugger view is included in the Programming and Debug Perspective (see page 24). To access the Snapshot Debugger view from any other perspective, select **Window → Show View → Other... → Achronix → Snapshot Debugger.**

This view allows running the Snapshot Debugger (see page 335) embedded in the design. A simple button press Collects Live Sample Data (see page 346) in a VCD file. This view also allows Configuring the Debug Capture Trigger Pattern(s) (see page 338), Configuring a Test Stimulus (see page 342), and Configuring the Data Capture Ranges (see page 344) before and after the trigger(s).

For convenience, favorite Snapshot configurations can be ( ) saved (see page 347) and ( ) loaded (see page 347) via the view toolbar buttons. Saved configurations may also be used to drive the Snapshot in Batch Mode (see page 348) via Tcl.

When a user design containing the `ACX_SNAPSHOT` macro completes the Flow Step (see page 221)**Run Prepare**, a `names.snapshot` configuration file, is automatically generated. This file contains harvested information from the design including the widths, depths and signal names for the monitor, trigger, and stimuli busses, user clock frequency, and default log and `.vcd` file path settings. When an Active Project and Implementation (see page 221) is available, the Snapshot View automatically loads the `names.snapshot` file to pre-populate the relevant fields of the view.

Note

When the `names.snapshot` file is generated, the file contains only a subset of a complete Snapshot configuration, and thus a generated `names.snapshot` file should not be used to drive Snapshot in Batch Mode (see page 348) via Tcl.

See also: Running the Snapshot Debugger (see page 335), Assign Bussed Values Dialog (see page 151), and Assign Bussed Signal Names Dialog (see page 149).

Table 87: *Snapshot Debugger View Toolbar Buttons*

Icon	Action	Description
	Arm Snapshot	Performs the following steps: 1. Sends the trigger conditions configuration to the Snapshot Debugger core. 2. Send the **Stimulus** value to the Design-Under-Test. 3. Waits for the trigger condition to be met. 4. Retrieves the trace buffer contents. 5. Outputs a VCD file. The Snapshot Debugger view runs the Achronix STAPL Player (`acx_stapl_player`) under the covers to control the JTAG interface.
	Cancel Snapshot	Cancels the Snapshot Arm by stopping the polling process and then resetting the `ACX_SNAPSHOT` macro.
	Save Snapshot Configuration	Saves the current settings of the Snapshot view to a text file. See Saving/Loading Snapshot Configurations (see page 347).
	Load Snapshot Configuration	Loads a previously saved configuration file. See Saving/Loading Snapshot Configurations (see page 347).
	Capture Snapshot Startup Trigger	Requires that the initial startup trigger parameters on the ACX_SNAPSHOT macro have been configured to enable the Startup Trigger feature, and that the Arm Snapshot action has not been executed since the bitstream has been programmed. Performs the following steps: 1. Waits for the startup trigger condition to be met. 2. Retrieves the trace buffer contents. 3. Outputs a VCD file.
	Configure JTAG Interface	Opens the preferences dialog with the Configure JTAG Connection Preference Page (see page 186) visible. See Configuring the JTAG Connection (see page 331).

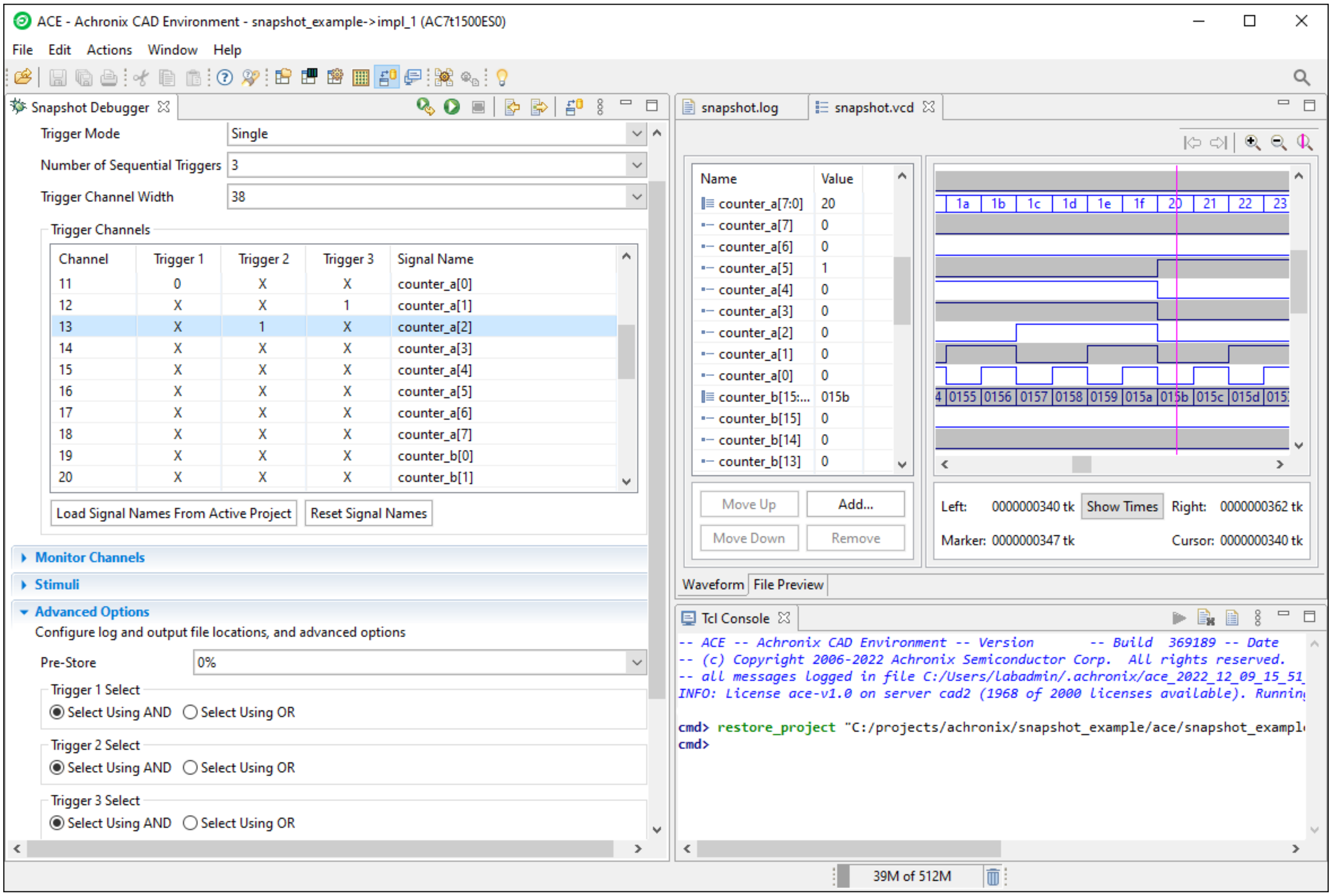


Figure 54: *Snapshot Debugger View Example*

Table 88: *Snapshot Debugger View Options*

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td colspan="2">Trigger Conditions</td></tr>
<tr><td>Trigger Mode</td><td>Allows selecting the trigger mode to use when the Arm Snapshot action is run.
The default trigger mode is Single:
1. The trigger conditions are programmed into the ACX_SNAPSHOT macro.
2. The GUI waits for a single trigger event to occur which matches those trigger conditions.
3. A single VCD file is recorded.
If Immediate trigger mode is selected, pressing the Arm button results in the same behavior as Single trigger mode, except that all 3 trigger patterns are treated as "Don't Care" (X) so that the trigger event occurs as soon as the Arm button is pressed.
If Repetitive trigger mode is selected:
1. The trigger conditions are programmed into the ACX_SNAPSHOT macro.
2. Samples are captured repetitively until the upper limit of trigger event records is reached.
When Repetitive trigger mode is selected, an additional set of repetitive trigger mode options appear allowing the configuration of:
• The number of sequential times Snapshot should be armed repetitively using the configured trigger conditions
• The way in which the output VCD files are managed</td></tr>
</table>

Option	Description
Number of Sequential Triggers	Allows selecting the use of either **1**, **2**, or **3** sequential triggers: • **1** – Trigger 2 and Trigger 3 are ignored during the match • **2** – Trigger 3 is ignored during the match and Snapshot triggers when Trigger 1 is matched, followed on any subsequent clock by a match on Trigger 2 • **3** – snapshot triggers after a match on Trigger 1, followed by Trigger 2, followed by Trigger 3 See Configuring the Trigger Pattern (see page 338), Configuring Test Stimulus (see page 342), and Configuring the Monitor Signals (see page 341).
Trigger Channel Width	The Snapshot debugger module can be configured to trigger channel widths of **1** to **40** channels. The Trigger Channel Width must be set to the value that corresponds with the configured Snapshot RTL instantiation. The trigger width is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Channel	The trigger channel number connected to the Snapshot Debugger core.
Trigger 1	Sets the Trigger 1 value for each channel. Valid options are: • X – don't care • R – rising edge • F – falling edge • 0 – level 0 • 1 – level 1 See Configuring the Trigger Pattern (see page 338).
Trigger 2	Sets the trigger 2 value for each channel. Valid options are: • X – don't care • R – rising edge • F – falling edge • 0 – level 0 • 1 – level 1 This column is only editable if 2 or 3 triggers are selected. See Configuring the Trigger Pattern (see page 338).
Trigger 3	Sets the trigger 3 value for each channel. Valid options are: • X – don't care • R – rising edge • F – falling edge • 0 – level 0 • 1 – level 1 This column is only editable if 3 triggers are selected. See Configuring the Trigger Pattern (see page 338).
Signal Name	Sets the user-defined name for the trigger channel. This signal name is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Load Signal Names From Active Project	When clicked, loads the `names.snapshot` file generated during design preparation (the **Run Prepare** flow step), which renames all signals with their project-specific names and loads other harvested project-specific settings.
Reset Signal Names	When clicked, renames all signals back to their default names, which is "signal" with a suffix corresponding to the channel number.
Repetitive Trigger Settings	
Record Limit	The repetitive trigger Record Limit setting determines how many times (number of records) the GUI repeatedly Arms the Snapshot debugger and captures samples. This may be set to automatically run Snapshot up to 128 times.

Option	Description
VCD Record Limit	Determines how many repetitively triggered Snapshot records to capture in a single VCD file. Essentially concatenates the VCD files from consecutive runs of Snapshot (records) into a single VCD file. The VCD file waveform contains a set of virtual signals to indicate the system timestamp at which each Snapshot record was captured. Up to 10 Snapshot records may be concatenated in a single VCD file.
Overwrite VCD File	When selected, the VCD Waveform File name specified in the Advanced Options section is used to store the output VCD file. The file is overwritten with the new VCD file each time the VCD record limit is reached. If not selected, multiple VCD files are written out and a unique VCD record number is added to the VCD Waveform File name specified in the Advanced Options section for each VCD. For example, if the Record Limit is set to **8**, the VCD Record Limit is set to **2**, and the VCD Waveform file path set to "./snapshot.vcd", Snapshot outputs 4 VCD files: 1. "./snapshot1.vcd" 2. "./snapshot2.vcd" 3. "./snapshot3.vcd" 4. "./snapshot4.vcd" Each file contains 2 Snapshot capture records.
Monitor Channels	
Monitor Channel Width	The Snapshot debugger module can be configured to monitor channel widths of 1 to 4087 channels. The Monitor Channel Width must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The monitor width is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Number of Samples	The Snapshot debugger module can be configured to capture between 512 and 16384 samples. The Number of Samples must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The number of samples is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Channel	The monitor channel number connected to the Snapshot Debugger core.
Signal Name	Sets the user-defined name for the monitor channel. This signal name is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited. This signal name is used in the VCD file waveform output.
Load Signal Names From Active Project	When clicked, loads the `names.snapshot` file generated during design preparation (the **Run Prepare** flow step), which renames all signals with their project-specific names and loads other harvested project-specific settings.
Reset Signal Names	When clicked, renames all signals back to their default names, which are "signal" with a suffix corresponding to the channel number.
Stimuli	
Stimuli Channel Width	The Snapshot debugger module can be configured to stimuli channel widths of 0 (no stimuli) to 512 channels. The Stimuli Channel Width must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The stimuli width is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Channel	The stimuli channel number connected to the Snapshot Debugger core.
Value	The value to drive out on this stimuli channel ARM_DELAY cycles before Snapshot is Armed (when the Arm button is pressed).
Signal Name	Sets the user-defined name for the stimuli channel. This signal name is automatically extracted from the user design and saved in the generated `names.snapshot` file, which can be loaded and edited.
Advanced Options	
Pre-Store	Controls the ratio of samples collected before and after the trigger. See Configuring Advanced Options (see page 344).

Option	Description
Trigger 1 Select	When set to **Select Using AND**, Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that **ALL** signal values not masked must match the specified pattern in order to generate a trigger match event. When set to **Select Using OR**, Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 338).
Trigger 2 Select	When set to **Select Using AND**, Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that **ALL** signal values not masked must match the specified pattern in order to generate a trigger match event. When set to **Select Using OR**, Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 338).
Trigger 3 Select	When set to **Select Using AND**, Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that **ALL** signal values not masked must match the specified pattern in order to generate a trigger match event. When set to **Select Using OR**, Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 338).
Frequency (MHz)	Must be configured to match the the `user_clk` timing constraint set in the SDC file of the design being debugged. This is automatically set according to the values captured in the `names.snapshot` file when an active implementation is available. See Configuring Advanced Options (see page 344).
File Paths Relative To	Chooses whether the **Log File** and **Waveform File** paths are understood to be relative to the **Active Project** directory or to the **Working Directory** (only matters when the file paths provided are relative paths, not absolute paths).
Log File	File name for the Snapshot log file, where raw Snapshot output (including warning and error messages) is logged. The default file name can be overwritten, and the accompanying **Browse** button may be used to graphically navigate to the desired directory or file. See Configuring Advanced Options (see page 344).
Waveform File	File name for the Snapshot VCD waveform output file, where the Snapshot sampled values (the trace buffer) is stored. The default file name can be overwritten, and the accompanying **Browse** button may be used to graphically navigate to the desired directory or file. See Configuring Advanced Options (see page 344).
Startup Trigger	This is the same as the ( ) **Capture Snapshot Startup Trigger** button in the view toolbar. See Collecting Samples of the User Design (see page 346).
Arm	This is the same as the ( ) **Arm Snapshot** button in the view toolbar. See Collecting Samples of the User Design (see page 346).
Cancel	This is the same as the ( ) **Cancel Snapshot** button in the view toolbar. See Collecting Samples of the User Design (see page 346).

Tcl Console View

The Tcl Console view provides an interactive Tcl console for ACE. All user interactions that change design and project data go through the Tcl command interface, including all commands executed while in the GUI. From here, executed commands and their information are displayed, including any warning and error messages. This console can also be used interactively by typing Tcl commands directly into the console to manipulate projects or the current design.

Valid ACE commands are highlighted in bold green. Informational messages are displayed in italic blue text. Warning messages are displayed in italic yellow text, and error messages are displayed in italic red text.

When the cursor is at the **cmd>** prompt, pressing the up arrow on the keyboard (↑) will move backward through the history of recently-issued commands to allow editing and re-issuing any prior command.

When typing in a command or filepath at the **cmd>** prompt, pressing the **TAB** key opens a dynamic content-assist list showing auto-completion candidates as well as command help text (if there are no valid choices available to complete the typing when TAB is pressed, a beep error tone sounds and no content-assist list appears). Pressing the up or down arrows on the keyboard moves through entries in the content-assist list, and pressing **Enter** chooses the selected entry in the list. Entries in the list may also be clicked with the mouse.

By default, the Tcl Console view is included in all Perspectives (see page 24). If it is not presently visible, to add it to the current perspective, select **Window** → **Show View** → **Tcl Console**.

For more details, see Using the Tcl Console (see page 301), check the available preferences on the Tcl Console View Preference Page (see page 210), and see the available commands in the Tcl Command Reference (see page 509).

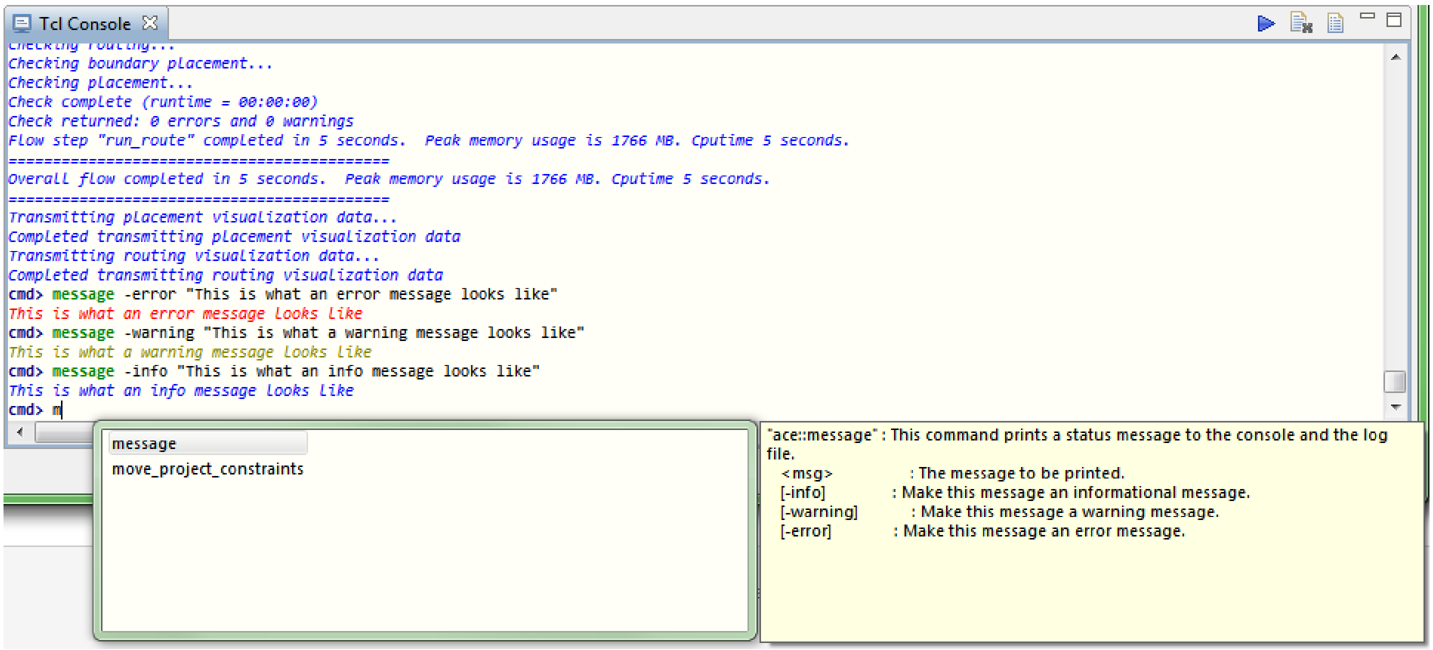


Figure 55: *Tcl Console View Example*

Table 89: *Tcl Console View Toolbar Buttons*

Icon	Action	Description
	Send command	Sends the current Tcl command at the console prompt. Alternatively, press **ENTER** on the keyboard to send the current command.
	Clear console	Clears the text in the console up to the current prompt line.
	Display log file	Opens the ACE log file for the current session in the Editor Area.

> **Warning!**
>
> When Tcl command return values are displayed in the Tcl Console, any long returned values are visually truncated at 500 characters in the console. The actual returned value is not edited, just the textual representation shown in the console. Thus, scripts using long return values still behave properly.

Dialogs

Several dialogs are used within ACE. These dialogs are typically shown in response to a user's specific menu choices or button presses.

Add Signals to Waveform Viewer Dialog

The Add Signals to Waveform Viewer Dialog appears when the **Add...** button is clicked in the VCD Waveform Editor (see page 28) editor.

This dialog allows:

1. Making signals visible which were previously hidden by clicking the **Remove** button in the VCD Waveform editor.
2. Adding duplicates of already-shown signals to the table and waveform.

Signals selected in the Add Signals to Waveform Viewer Dialog may be inserted after the first row currently selected in the VCD signal table, or at the top of the table if no rows are selected. The selected signals are added or inserted using the **Append** and **Insert** buttons. The **Move Up** and **Move Down** buttons on the VCD Waveform editor may be used to move the selected signals from wherever they are initially inserted or appended.

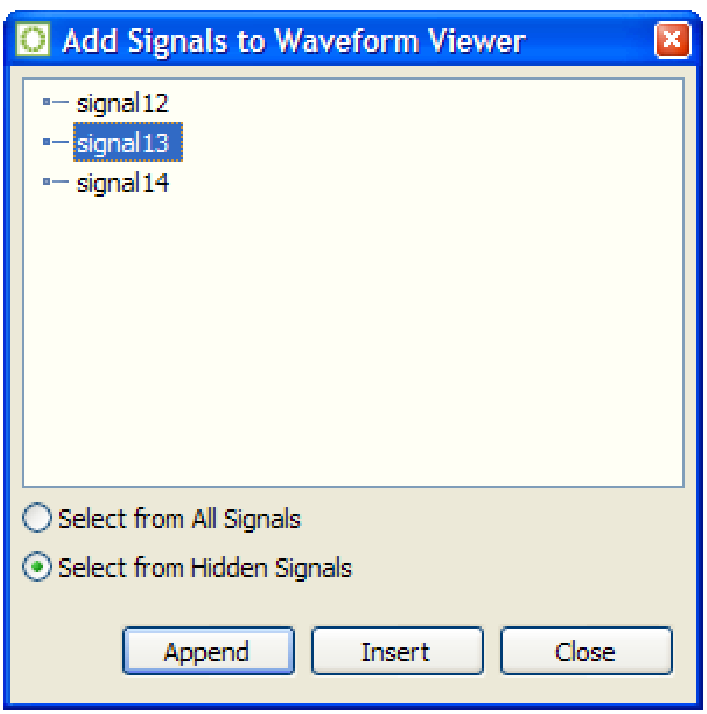


Figure 56: ***Add Signals to Waveform Viewer Dialog Example***

The majority of the dialog is taken up by an area listing the signals. The listed signals vary depending upon the radio-button currently selected in the dialog.

Table 90: *Add Signals to Waveform Viewer Dialog Actions*

Action	Description
Select from All Signals	When selected, causes the list to be populated with all signals contained in the current VCD file.
Select from Hidden Signals	When selected, causes the list to be populated with all signals found in the VCD file which are currently hidden (signals removed from the VCD Waveform editor signal table are considered hidden). If no signals are currently hidden, the list of hidden signals is empty.
Append	Appends the currently-selected signal to the bottom of the VCD Waveform Editor signal table.
Insert	Inserts the signal currently selected in the dialog list below the signal currently selected in the VCD Waveform Editor signal table. If no signal was selected in the VCD Waveform Editor signal table when the **Add...** button was pressed to bring up this dialog, the signal selected in the dialog is inserted at the top of the VCD Waveform Editor signal table.
Close	Closes the dialog.

Note

- The **Append** and **Insert** buttons may each be clicked multiple times for a given signal, which adds the signal selected in the dialog list to the VCD Waveform Editor signal table multiple times.
- These buttons are disabled if no signal is currently selected in the dialog signal list.
- If either button is used to un-hide a previously-hidden signal, the signal is removed from the list of hidden signals since it is no longer considered hidden.

There are also some icons used by content displayed in the dialog signal list as shown below.

Table 91: *Add Signals to Waveform Viewer Dialog Icons*

Icon	Description
	Signal
	Bus

Add Source Files Dialog

The Add Source Files dialog is used to browse for netlist (`.v`, `.vm`, and `.vma`), constraints (`.sdc` and `.pdc`), and IP Configuration (`.acxip`) Source Files (see page 217) to add to the selected Project (see page 215). After selecting the files to add, click **Open** (in Windows) or **OK** (in Linux) to add them to the project.

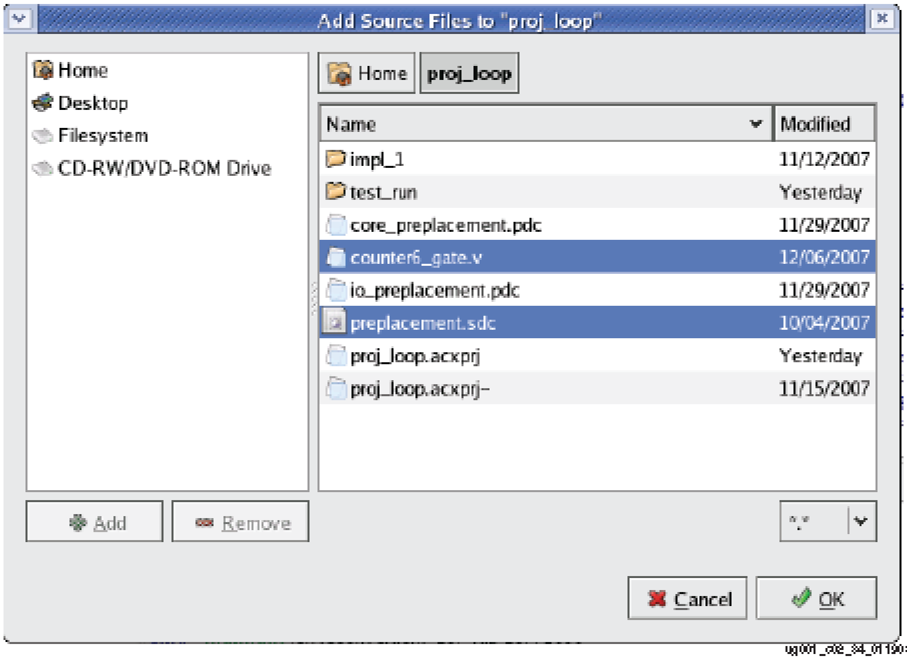


Figure 57: ***Add Source Files Dialog Example***

> **Note**
>
> ACE loads source files in the same order they were added to the project. If ACE is loading files in an incorrect order:
>
> 1. Remove all source files from the project.
> 2. Add files, *one at a time*, to the project in the desired order.

When files with unrecognized file extensions are added to a project (possible when the "*.*" file filter is selected in the Add Source Files dialog), a second dialog appears requesting the categorization of the unknown file extensions.

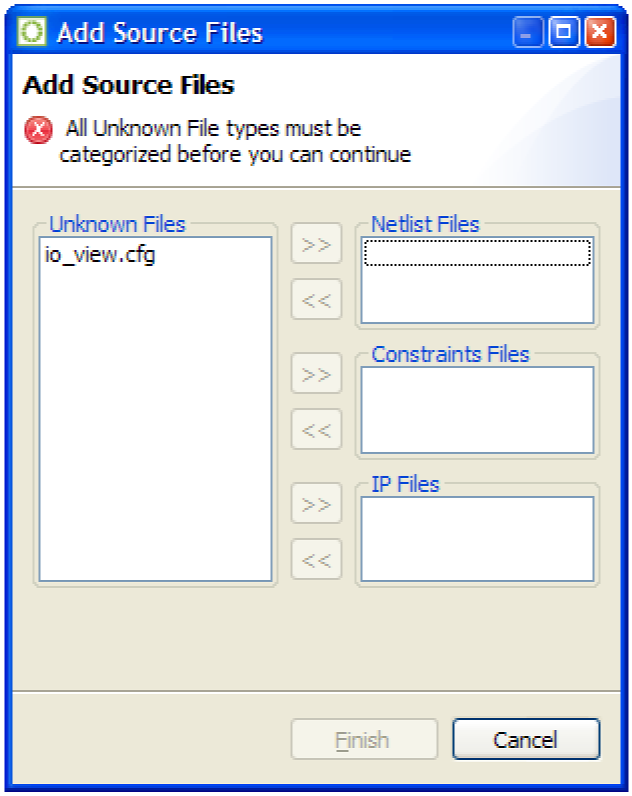


Figure 58: ***Add Source Files Categorization Dialog Example***

The categorization dialog contains the list of unknown files on the left, with the allowed categories for each file on the right. Files may be moved into and out of the categories with the **>>** and **<<** buttons, respectively.

When all the files are categorized, click the **Finish** button to add the files to the active ACE project, or click Cancel to add none of the files.

See also: Adding Source Files (see page 272) and Adding Configuration Files to a Project (see page 309).

Assign Bussed Signal Names Dialog

The Assign Bussed Signal Names Dialog wizard allows the assigning multiple signal names from the SnapShot Debugger view (see page 139) "Monitor Channels", "Trigger Channels", or "Stimuli Channels" tables using bus notation. After configuring the bus in the dialog, the bus name and indices are propagated to all the selected signals, changing the signal names appropriately. Monitor channel signal names are then used in the SnapShot sampled output, visible in the VCD Waveform Editor (see page 28).

> **Note**
>
> This dialog is only useful when creating a Snapshot configuration from scratch. Typically, this dialog is not needed since ACE automatically outputs all the signal names from the user design into the `names.snapshot` file as part of the normal ACE Place and Route flow.

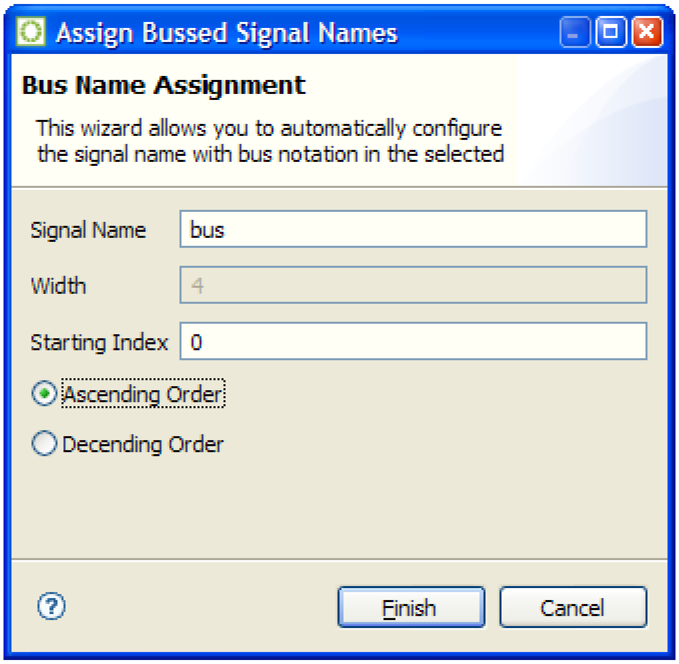


Figure 59: *Assign Bussed Signal Names Dialog Example*

Table 92: *Assign Bussed Signal Names Dialog Options*

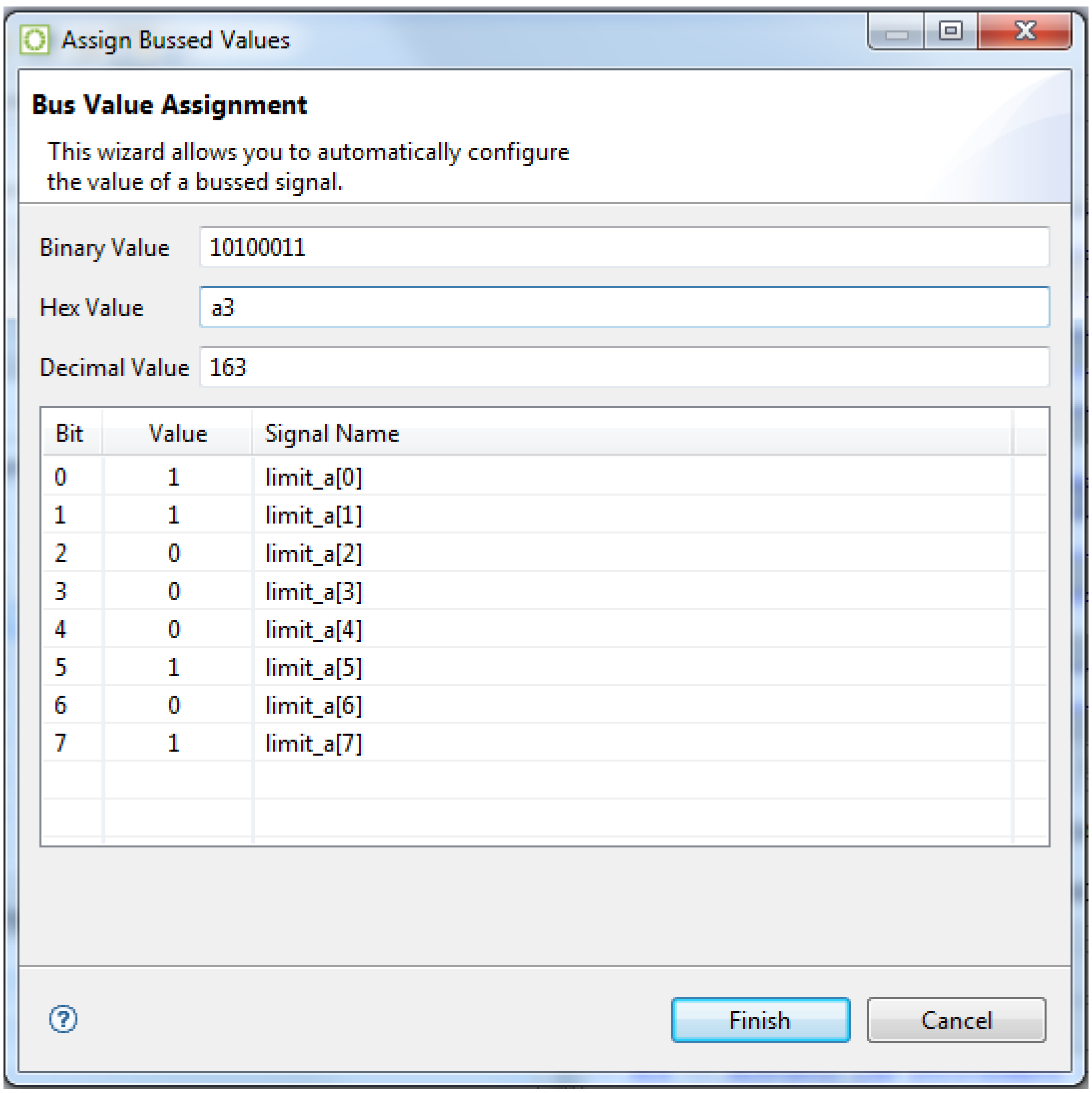
Option	Description
Signal Name	The desired name of the bus.
Width	The width (in bits) of the bus. This value is not editable. It reflects the number of signals which were selected from the table in the Snapshot Debugger View (see page 139).
Starting Index	The desired starting index of the bus, sometimes also called the offset into the bus.
Ascending Order	When selected, the bus indices start at **Starting Index** and increment **Width** times.
Descending Order	When selected, the bus indices start at **Starting Index** and decrement **Width** times.
Finish	Accepts the specified bus configuration, closes the dialog, and applies the changes to the SnapShot Debugger view (see page 139) table.
Cancel	Discards the specified bus configuration and closes the dialog. No changes are applied to the SnapShot Debugger view (see page 139) table.

Assign Bussed Values Dialog

The Assign Bussed Values Dialog allows assigning a value to multiple signals from the SnapShot Debugger view (see page 139) "Trigger Channels" or "Stimuli Channels" tables as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all the selected signals in the SnapShot Debugger View (see page 139). There are two ways to launch this dialog to allow bus assignment of values:

1. Click to select a single row in the SnapShot Debugger View (see page 139) table which has a bussed signal name (i.e., `din[2]`). Then, right click to edit the **Value by Bus**. This method automatically finds all other bits in the bus with the same signal name (e.g., `din[0]`, `din[1]`, `din[2]`, etc.) and opens the dialog to allow editing of the entire bus of signals.
2. Hold CTRL or SHIFT and click to select multiple rows in the SnapShot Debugger View (see page 139) table. Then, right-click to edit the **Value by Selection**. This method opens the dialog to allow editing of all selected signals as a bussed value.

See also: Configuring the Trigger Pattern (see page 338).



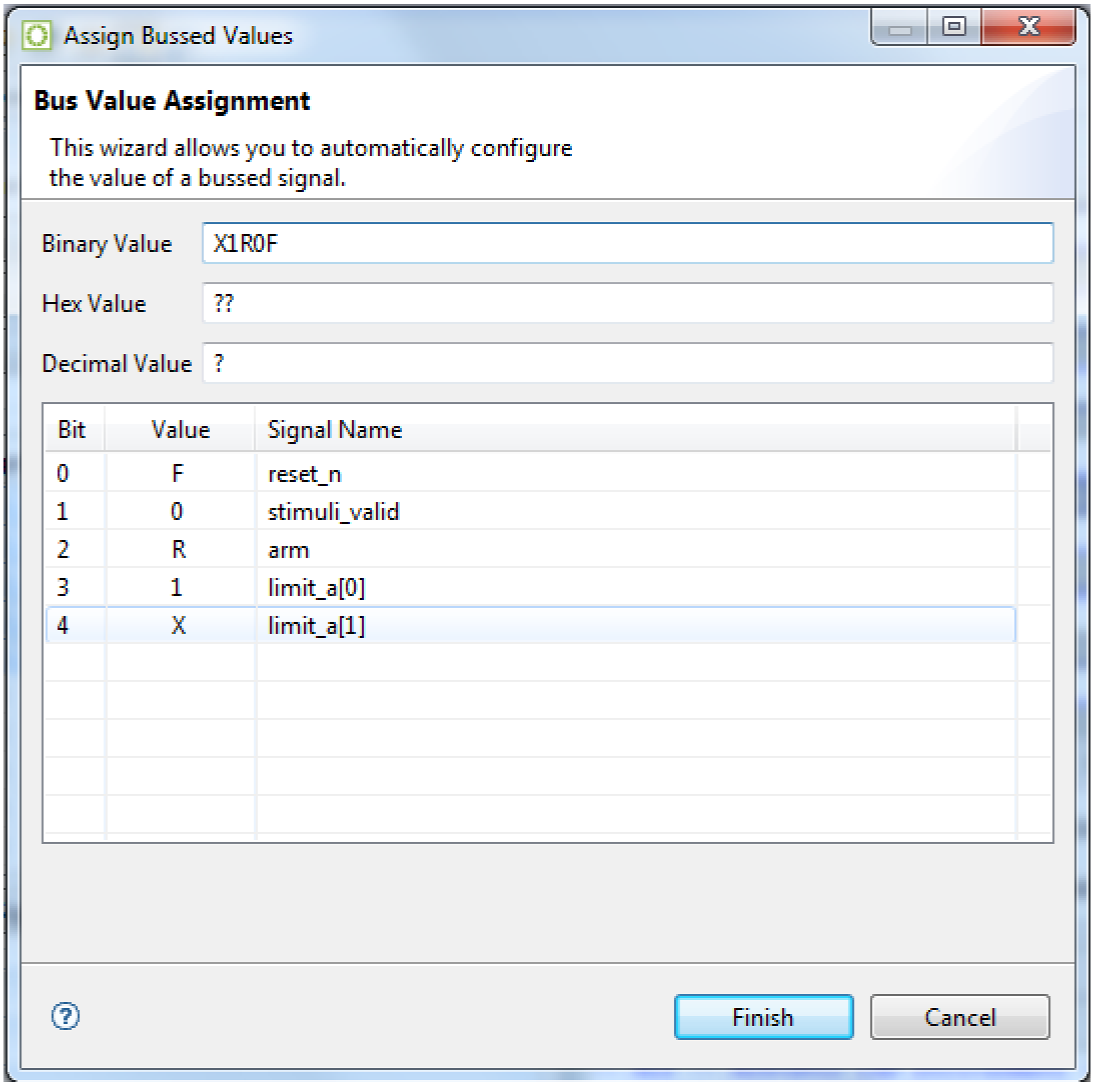


Figure 60: ***Assign Bussed Values Dialog Examples***

Table 93: ***Assign Bussed Values Dialog Options***

Option	Description
Binary Value	The desired value for the bus in binary. Valid values for each bit for Trigger Channels are **X** (don't care), **R** (rising edge), **F** (falling edge), **1** (level 1), and **0** (level 0). Valid values for each bit for Stimuli Channels are **1** (level 1), and **0** (level 0). The right-most bit corresponds to bit 0 in the table of signal names, and the left-most bit corresponds to the MSb in the table.
Hex Value	The desired value for the bus in hexadecimal. This field is only capable of representing level (1 or 0) values for each channel. **X** (don't care), **R** (rising edge), and **F** (falling edge) binary values result in a "?" character in this field.
Decimal Value	The desired value for the bus in decimal. This field is only capable of representing level (1 or 0) values for each channel. **X** (don't care), **R** (rising edge), and **F** (falling edge) binary values result in a "?" character in this field.

Option	Description
Bit	The bit offset into the bus value being edited.
Value	The bit value at the bit offset into the bus value being edited.
Signal Name	The signal name at the bit offset into the bus value being edited.
Finish	Accepts the specified bus configuration, closes the dialog, and applies the changes to the corresponding SnapShot Debugger view (see page 139) table.
Cancel	Discards the specified bus configuration and closes the dialog. No changes are applied to the corresponding SnapShot Debugger view (see page 139) table.

Configure Clock Pre-Routes Dialog

The Configure Clock Pre-Routes Dialog appears after the **Configure Clock Pre-Routes...** action is selected from a context menu in the Clock Regions View (see page 36), Clusters View (see page 41), Partitions View (see page 117), or Placement Regions View (see page 120). This dialog allows the creation of new clock pre-route constraints.

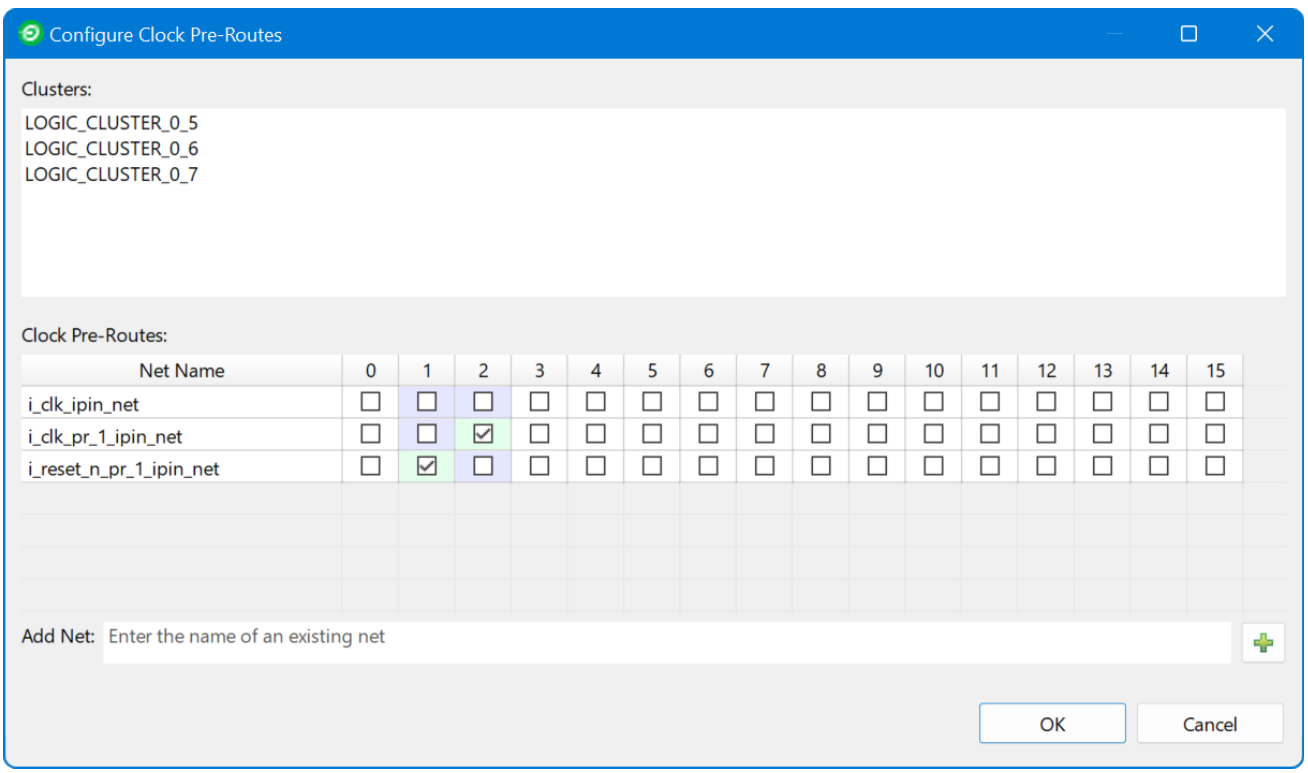


Figure 61: ***Configure Clock Pre-Routes Dialog Example***

Table 94: *Configure Clock Pre-Routes Dialog*

Option	Description
Clock Regions/Clusters /Partitions/Placement Regions	A list of the targets to which pre-route table changes are to be applied.
Net Name	Each row in the table contains the name of a net to be constrained.
Numbered columns	Each numbered column represents a numbered clock track. Checking a cell in the table constrains the given net to the given clock track.
Add Net	Type the name of any net in the design into the text field, press ENTER or click the ( ) **Add** button to add that net to the table.

The table in this dialog uses background colors to indicate the actions performed since the dialog appeared:

- A red background indicates that a check box was removed. Red cells indicate that one or more `remove_clock_preroute` commands are to be issued if the **OK** button is clicked.
- A green background indicates that a check box was added. Green cells indicate that one or more `add_clock_preroute` commands are to be issued if the **OK** button is clicked.
- A purple background indicates that a check box exists somewhere in the given clock track column. Only one net at a time may be associated with any clock track. Checking any cell in the table automatically unchecks all other cells in that column.
- Hovering over a green or red cell shows the `add_clock_preroute` or `remove_clock_preroute` commands to be issued for that row in the table when the **OK** button is clicked.

> **Note**
>
> If some, but not all, targets have a given net associated with a given clock track, *all* targets can be assigned that net/clock track association by unchecking and then re-checking the appropriate check box. When the dialog was first shown, the check box had a purple background. After unchecking and then re-checking, the box has a green background to indicate that new `add_clock_preroute` commands are to be executed to cover any additional targets.

For example, in the following example, the previous example was changed as follows:

- (`i_reset_n_pr_1_ipin_net` : 1) was clicked, unchecking it
- (`i_clk_ipin_net` : 2) was clicked, checking it and automatically unchecking (`i_clk_pr_1_ipin_net` : 2)
- (`i_clk_pr_1_ipin_net` : 4,5,6) were all clicked, checking them

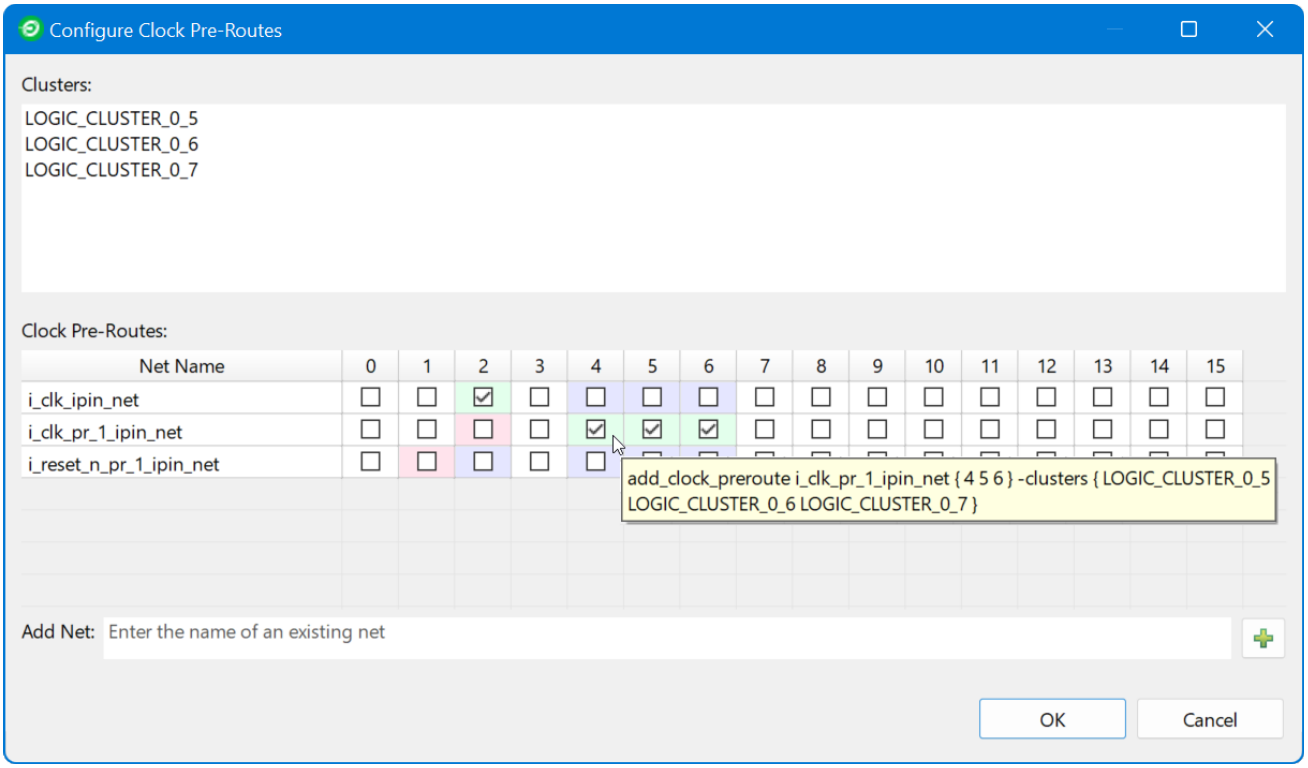


Figure 62: *Configure Clock Pre-Routes Dialog Update Indicators*

Configure Table Columns Dialog

The Configure Table Columns Dialog allows configuring the columns shown in the active view. Currently this dialog is only available for the IO Assignment View (see page 74). From the dialog, the columns which are visible and the width (in pixels) of each column may be configured. The current column configuration may also be saved to a file, or a previous column configuration may be loaded from a file.

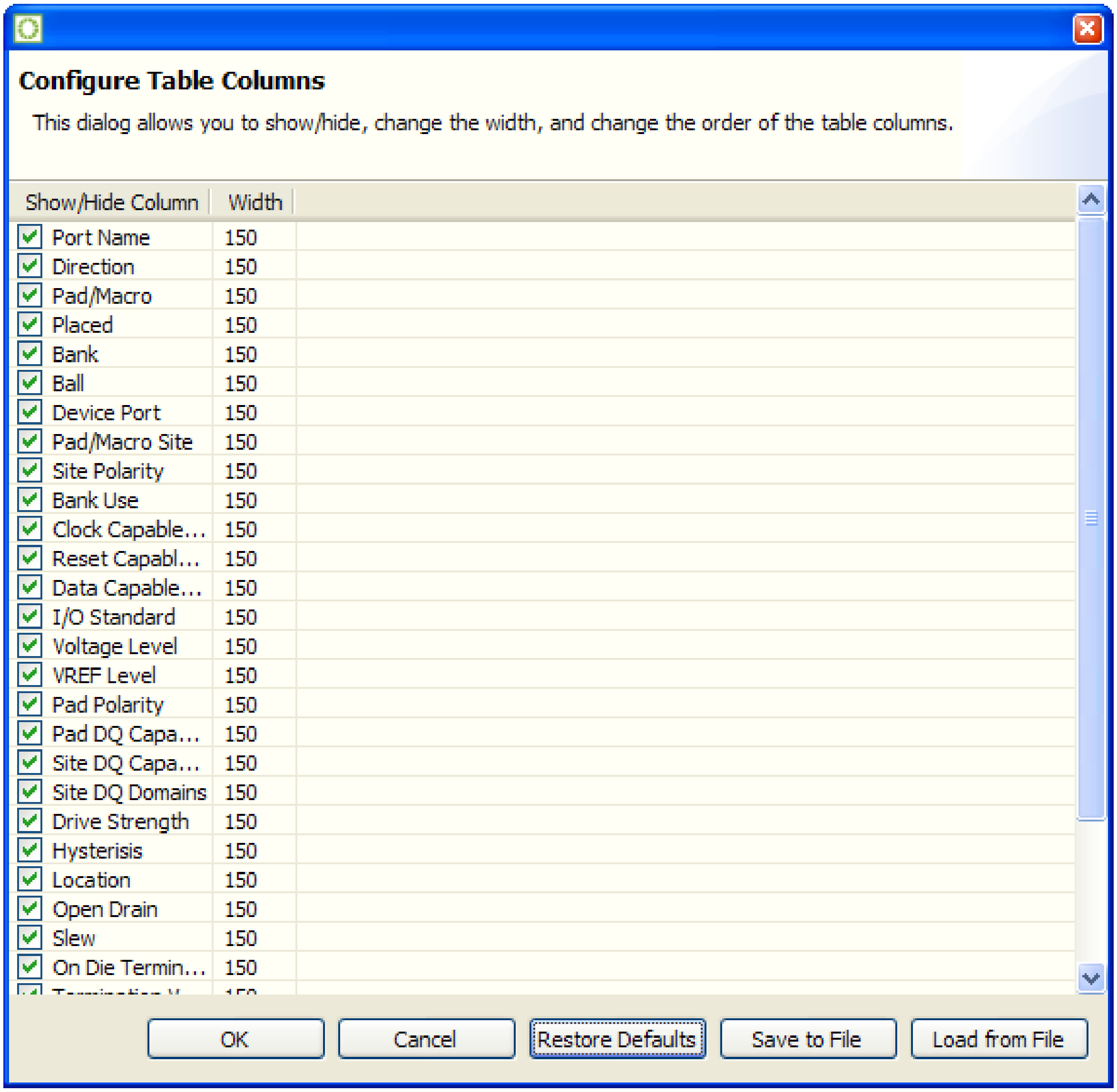


Figure 63: ***Configure Table Columns Dialog Example***

Create a New Constraints File Dialog

The Create a New Constraints File Dialog is used to easily create a new, empty constraints file and optionally add it to the currently active project. The dialog is available in all perspectives, and can be accessed by selecting **File** → **New** → **SDC Constraints File…**.

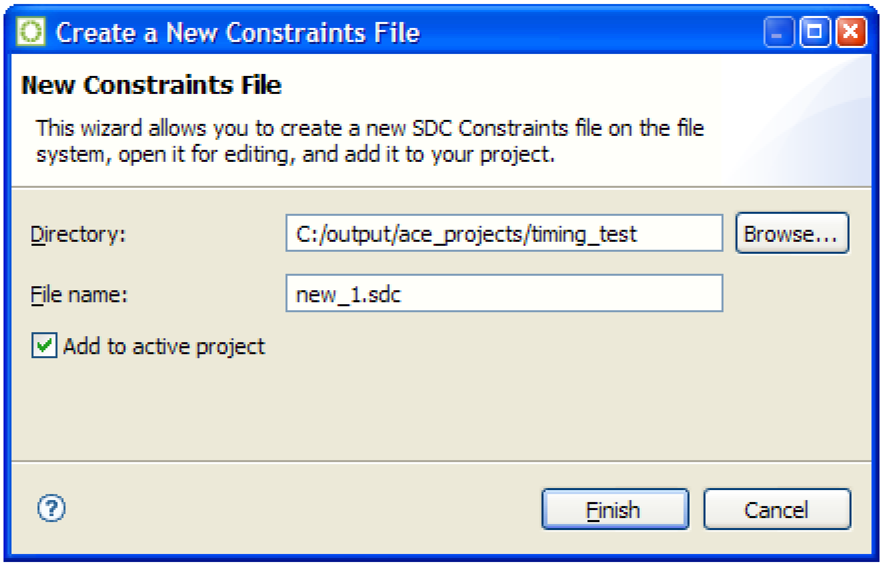


Figure 64: *Create a New Constraints File Dialog Example*

The dialog allows typing the file destination Directory, or selecting it graphically using the **Browse…** button. The Directory name provided must already exist. (If selected, the **Browse...** button displays a Directory Selection Dialog, which also allows creating a new directory and then selecting it.)

The File Name must be unique — a file with that name must not already exist in the destination Directory.

If there is currently an active project in ACE, the **Add to active project** checkbox will be enabled and checked by default. If there is no project active, the checkbox will be disabled and deselected.

When **Finish** is selected, the text file is created, and a Text Editor (see page 28) is opened in ACE for the new text file.

Note

This dialog may be used to create PDC files as well as SDC files. Simply use `.pdc` instead of `.sdc` for the file extension.

Create a New Text File Dialog

The Create a New Text File Dialog simply allows creating a new text file and opening the file in the ACE text editor in a single action. The dialog is available in all Perspectives (see page 24), and can be selected via **File** → **New** → **Text File**
.

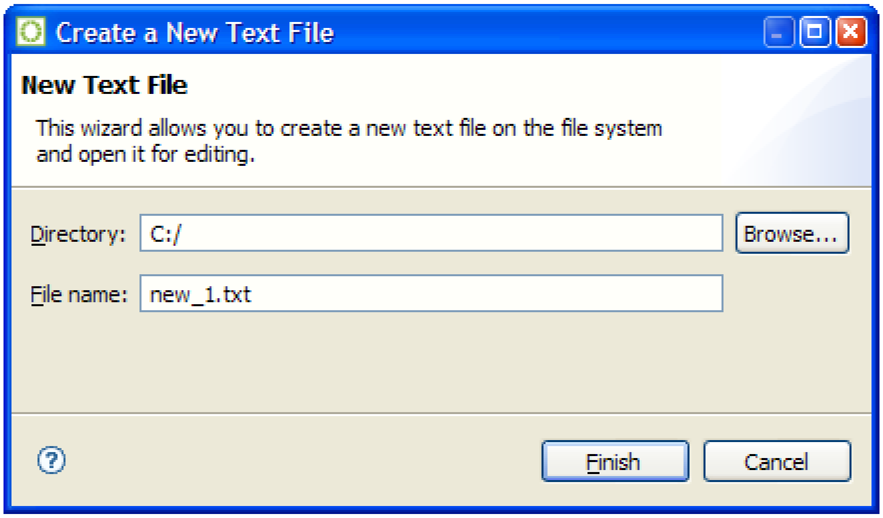


Figure 65: ***Create a New Text File Dialog Example***

The dialog allows typing the file destination directory, or selecting it graphically using the **Browse...** button. The **Directory** name provided must already exist (if selected, the **Browse...** button displays a Directory Selection Dialog, which also allows creating a new directory and then selecting it).

The **File name** must be unique — a file with that name must not already exist in the destination Directory.

When **Finish** is selected, the text file is created, and the ACE Text Editor (see page 28) is opened for the new text file.

Create Implementation Dialog

The Create Implementation Dialog is used to create a new implementation in the selected project. After indicating a new name for the implementation and whether to copy option values from the active implementation, click **Finish** to create the implementation in the selected project.

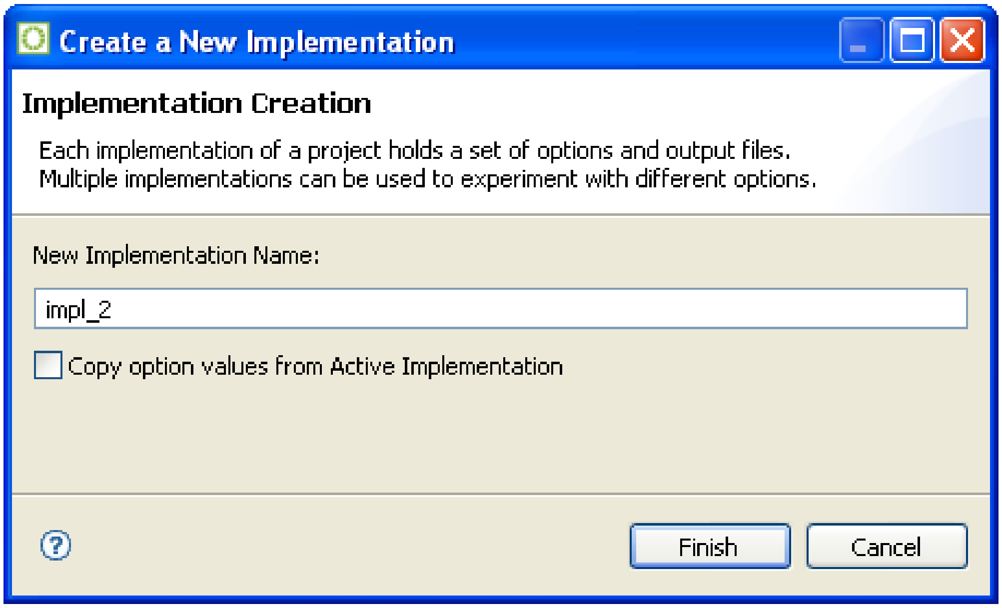


Figure 66: ***Create Implementation Dialog Example***

Table 95: ***Create Implementation Dialog Fields***

Field	Default	Description
New Implementation Name	`impl_`	The name of the new project implementation to be created. This name must be unique among existing implementations in the selected project. The new name is used to create a new directory under the project directory for the selected project.
Copy Option Values from Active Implementation	Unchecked	If this field is checked, option values from the current Active Implementation are copied into the new implementation.

Create Placement Region Dialog

This wizard dialog appears after a drag-and-drop to define a rectangular area in the Floorplanner View (see page 53) while the Placement Region Tool is active. This dialog allows naming the new Placement Region, and defining its bounds.

See also: Creating a New Placement Region (see page 355).

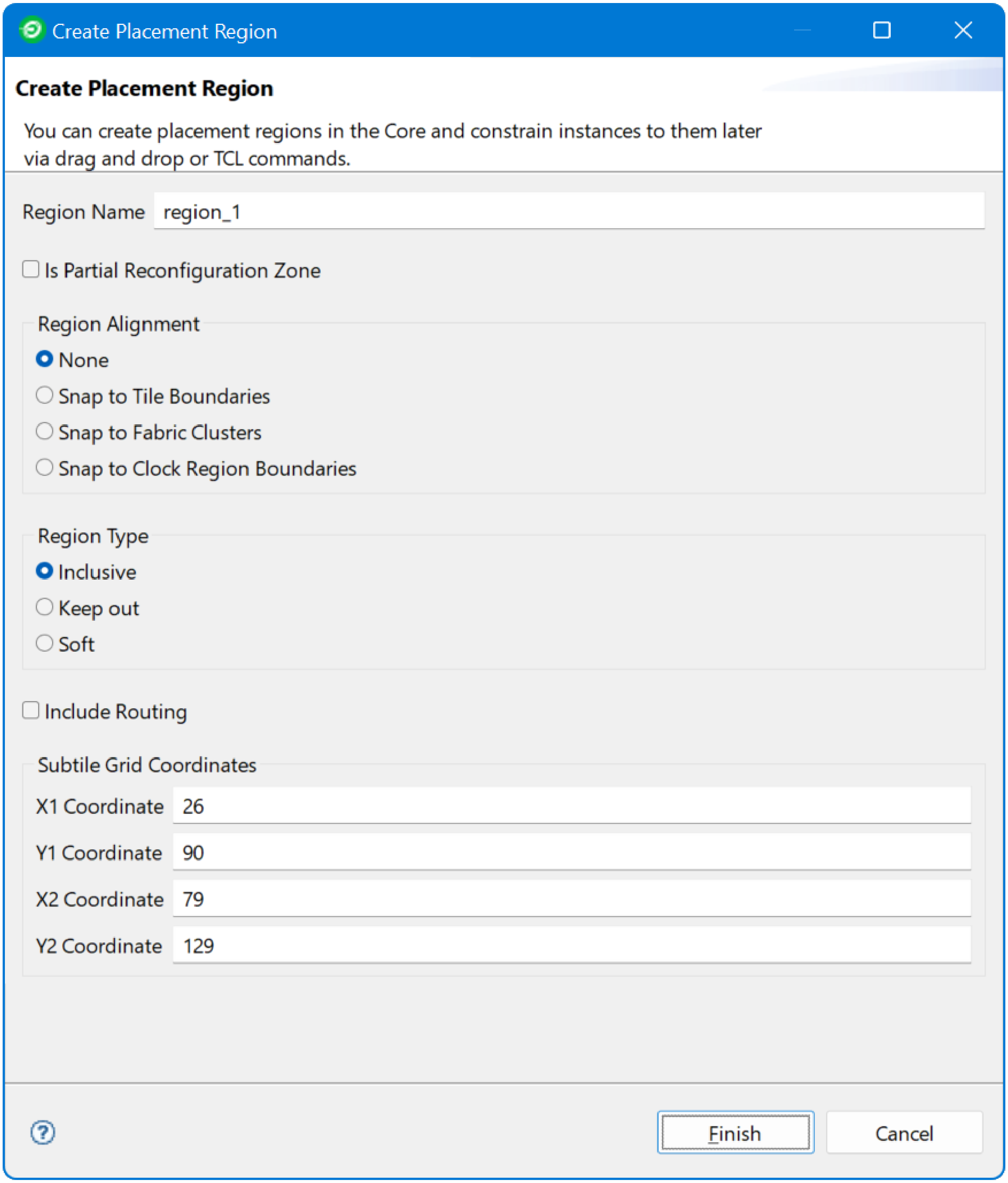


Figure 67: *Create Placement Region Dialog Example*

Table 96: *Create Placement Region Dialog Options*

Option	Description
Region Name	The name for the new Placement Region. ACE pre-populates this field with a default incrementing value.
Include Routing	Treats the region as a routing constraint as well as a placement constraint, keeping all routing wires and instances inside the region boundary box.
Is Partial Reconfiguration Zone	Indicates the region is intended to be used for partial reconfiguration.
Region Alignment	
None	No snapping.
Snap to Tile Boundaries(1)	If selected, ACE creates a Placement Region that encompasses all Tiles selected within the drag-and-drop rectangle.
Snap to Fabric Clusters(2)	If selected, ACE creates a Placement Region that encompasses all Fabric Clusters within the drag-and-drop rectangle.
Snap to Clock Region Boundaries(3)	If selected, ACE creates a Placement Region that encompasses all Clock Regions which contain any of the selected Tiles.
Region Type	
Inclusive	Instances added to the region are placed within the region bounding box. Permits instances to be placed inside the region even if they do not belong to the region.
Keep out	Prevents any instances from being placed inside the region. No instances may be added to a **Keep Out** region.
Soft	Instances added to the region are pulled toward the region center during placement, but instances are permitted to overflow the bounds of the soft region. Soft Placement Regions are rendered as ellipses in the Floorplanner View (see page 53), and the center of the ellipse acts as a center-of-gravity for placement. Soft regions do not limit the contained number of constrained instances, nor do they have a true count of contained sites of each resource. See Placement Regions and Placement Region Constraints (see page 354) for more details.
Subtile Grid Coordinates	
X1 Coordinate	The upper-left X coordinate within the subtile grid, corresponds to the left edge.
Y1 Coordinate	The upper-left Y coordinate within the subtile grid, corresponds to the top edge.
X2 Coordinate	The lower-right X coordinate within the subtile grid, corresponds to the right edge.
Y2 Coordinate	The lower-right Y coordinate within the subtile grid, corresponds to the bottom edge.

Table Notes

1. Since Placement Regions can only contain entire sites (no partial sites), the Placement Region can potentially grow larger than the outline rectangle.
2. In this mode, since Placement Regions can only contain entire Fabric Clusters (no partial Fabric Clusters), the Placement Region almost certainly grows larger than the outline rectangle.
3. If selected, ACE creates a Placement Region that encompasses all Clock Regions which contain any of the selected Tiles.

Note

When using Subtile Grid Coordinates, the 0,0 coordinate maps to the upper-left of the Core+Boundary in the Floorplanner view. The exact coordinates of the lower-right corner coordinate limits of the Core+Boundary vary by device.

Create Project Dialog

The Create Project Dialog helps create a new project in the Workbench. After indicating a name and location for the project, click **Finish** to create the project.

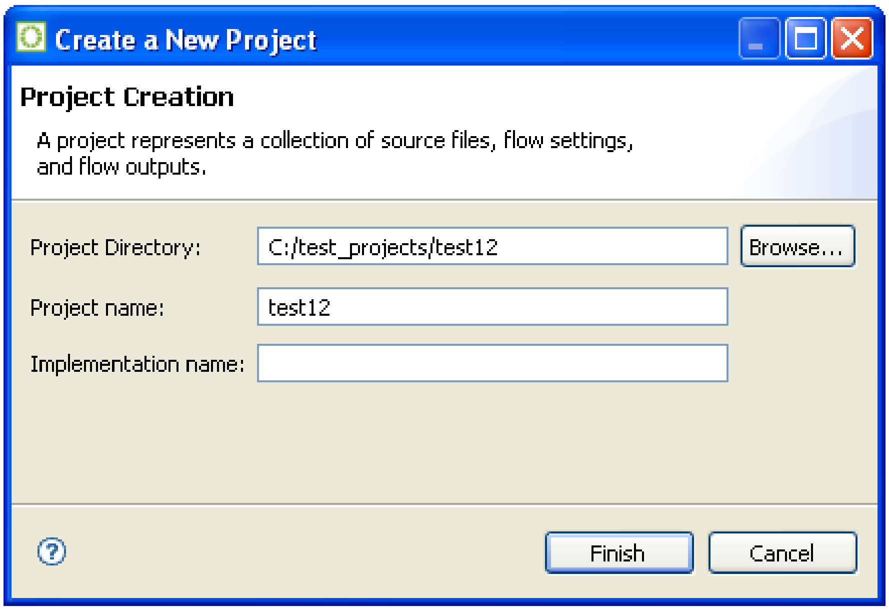


Figure 68: ***Create Project Dialog Example***

Table 97: ***Create Project Dialog Example***

Field	Description
Project Directory	The location in the file system where the project is created. Either type the new location or browse to select a file system location for the new project.
Project Name	The name of the new project to be created. This name is the base name of the `.acxprj` file created in the project directory.
Implementation Name	The name of the new implementation to be created with the project. Leaving this blank causes a name to be chosen automatically.

Generate a Pin Assignment Report Dialog

The Generate a Pin Assignment Report Dialog allows generating a customized Pin Assignment report (see page 227) with a column organization identical to that of the IO Assignment view (see page 74).

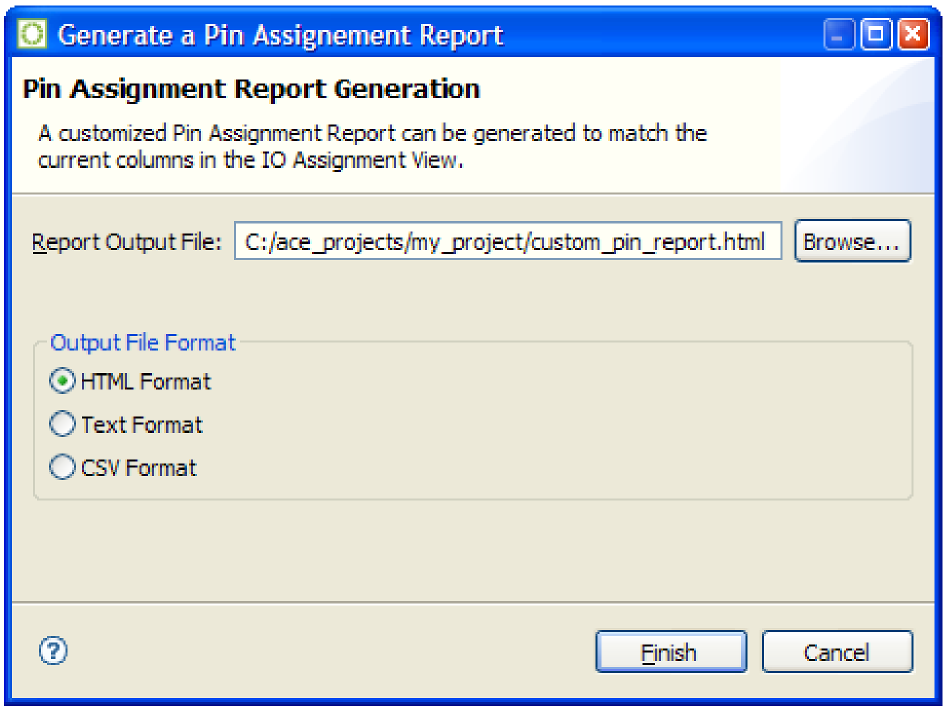


Figure 69: ***Generate a Pin Assignment Report Dialog Example***

Generate IP Design Files Dialog

The Generate IP Design Files Dialog is used to create the necessary RTL models, timing constraints and bitstream files for configuring embedded IP. The files generated are based upon the configuration file (`.acxip`) created via the active IP Configuration Editor (see page 26).

See also: Creating an IP Configuration (see page 306).

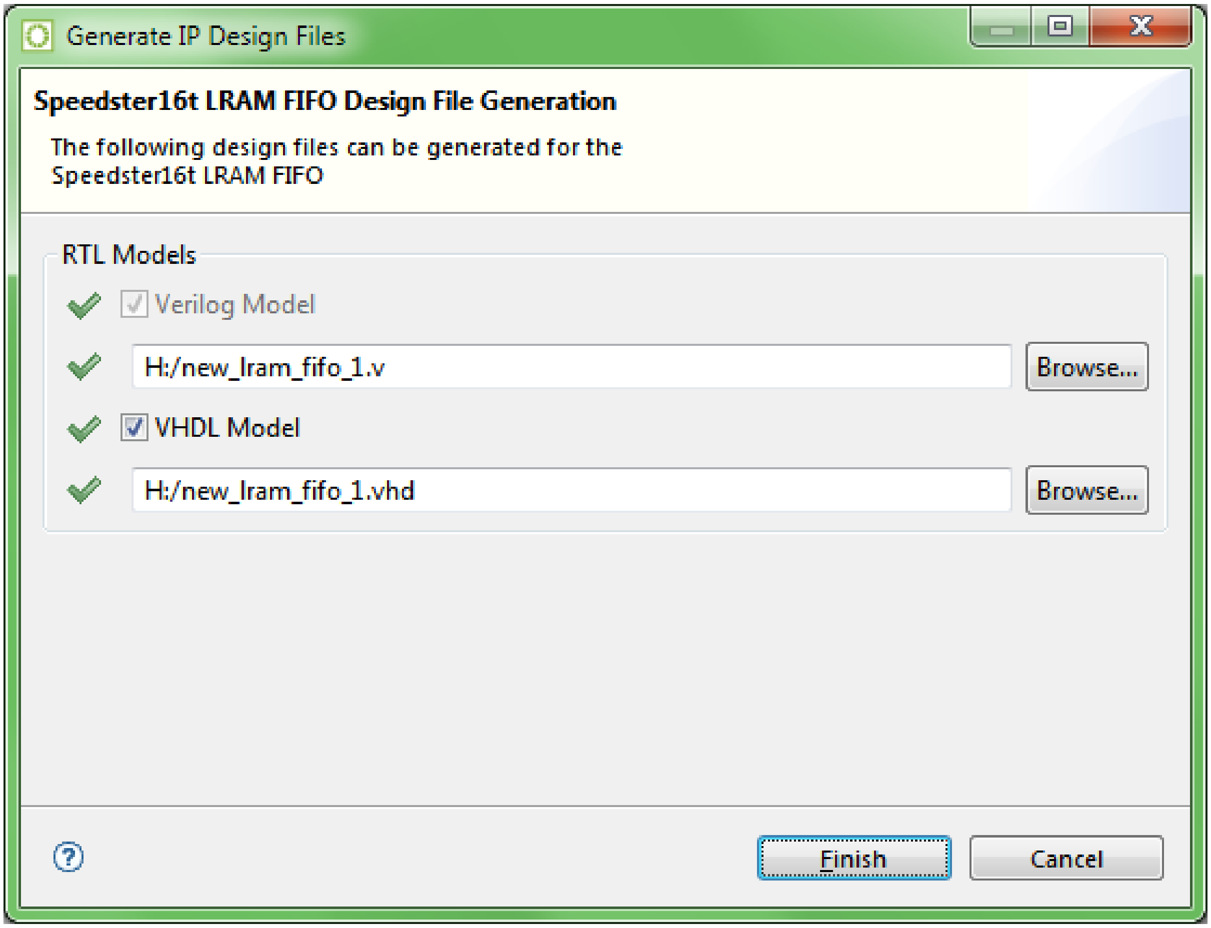


Figure 70: ***Generate IP Design Files Dialog Example***

> **Note**
>
> Each IP Configuration Editor has its own set of output files which are specific to the type of IP being configured. For example, some types of IP require PDC or SDC constraints files, while other types of IP do not. The table of dialog field descriptions below describes the common output files for most types of IP.

Table 98: ***Generate IP Design Files Dialog Fields***

Field	Default	Description
RTL Models		
Verilog Model(1)	Selected	Selects whether a Verilog model for the configuration is generated.
VHDL Model(1, 2)	Deselected	Selects whether a VHDL model for the configuration is generated.
Timing Constraints		
SDC Constraints(1)	Selected	Selects whether an SDC constraints file for the configuration is generated.

Field	Default	Description
Placement Constraints[1]	Selected	Selects whether a placement constraints file for the configuration is generated.

Table Notes

1. The default file path is displayed below the option. An alternate path can be selected via the Browse button.
2. The VHDL RTL Model is a simple wrapper around the Verilog RTL Model. Because of this, when using the VHDL RTL Model, the Verilog RTL Model must also be generated and included in the user design.

Load Project Dialog

The Load Project Dialog allows browsing to find an existing project file to load into the Workbench. After selecting the project file and choosing to activate an implementation, click **Finish** to load the project.

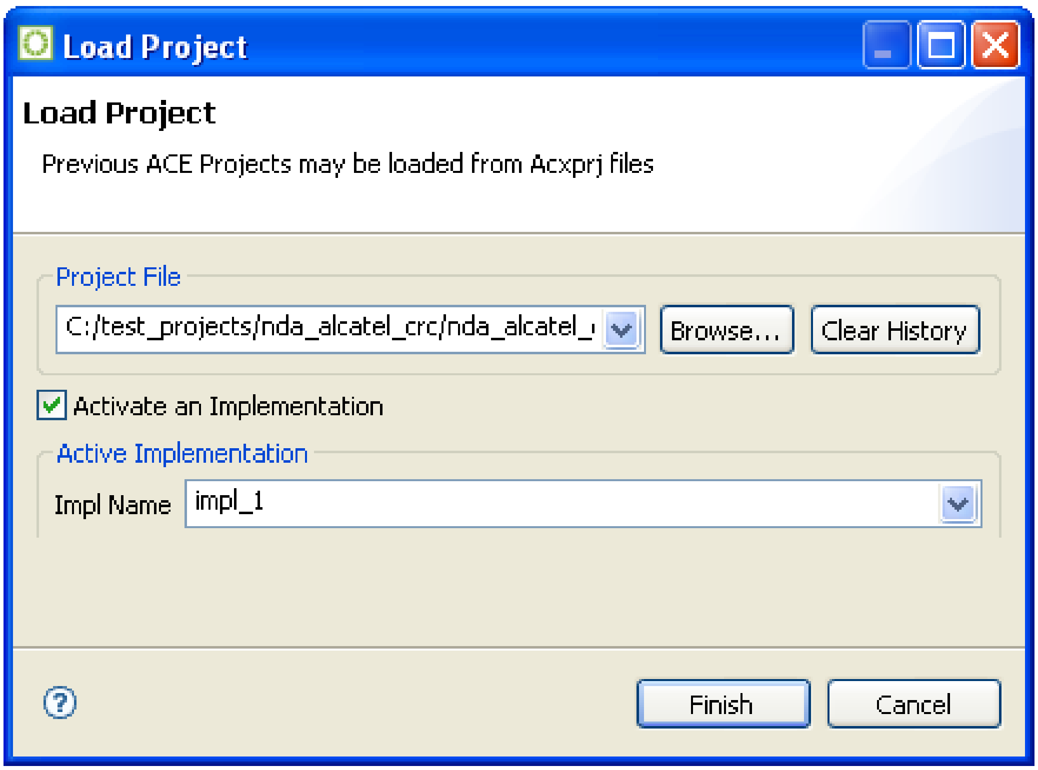


Figure 71: ***Load Project Dialog Example***

Table 99: ***Load Project Dialog Fields***

Field	Description
Project File	The path to the ACE Project File (`.acxprj`) to load. Either type the new location or browse to select a file system location for the project. A history of previously loaded projects can be accessed via the drop-down list. This previous project history may be cleared at any time by clicking the **Clear History** button.
Activate an Implementation	Choose to activate an implementation upon loading the project. If another project is already loaded, this field can be un-checked to preserve the current active implementation in the ACE session.

Field	Description
Impl Name	The name of the implementation to activate after loading the project. A drop-down list allows selecting from any implementation defined within the specified project file.

New IP Configuration Dialog

The New IP Configuration Dialog helps create a new IP configuration file (`.acxip`). After indicating a name and location for the configuration file, click **Finish** to create the file and open the relevant IP Configuration Editor (see page 26). See also: Creating a New IP Configuration (see page 306).

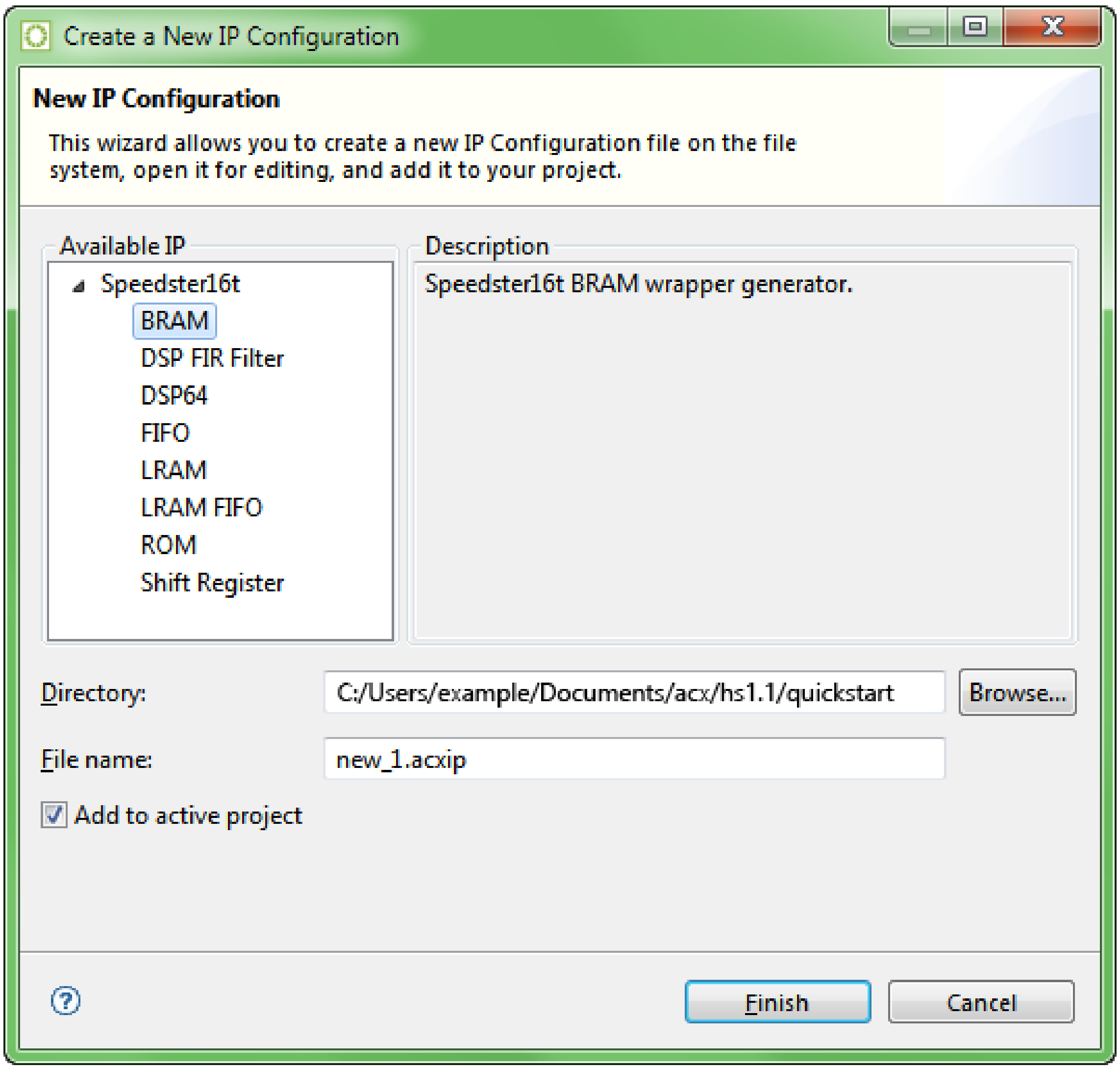


Figure 72: ***New IP Configuration Dialog Example***

> **Note**
>
> The displayed IP Libraries and IP types are dynamic and change based on which technology libraries and devices are installed and licensed. The screenshots and example descriptions in this section do not necessarily reflect the IP types of the actual device being used.

New IP Configuration Dialog Fields

Field	Default	Description
Available IP	–	Lists the available IP blocks by FPGA family.
Description	–	Provides a description of the IP block.
Directory	Current active project directory	The location in the file system where the configuration file is created. Either type the new location or browse to select a file system location for the new configuration.
File Name (1)	`new_.acxip`	The name of the new configuration file to be created. The file name (without the `.acxip` suffix) also becomes the module name in the generated IP.
Add to active project	Enabled	This check box allows the IP configuration file to be added to the current project (this option is only available if a project is active). If unchecked, the IP configuration file is created at the chosen path but not automatically added to any project. Because it is not a member of a project, the new `.acxip` file is not visible in the Projects view.

Table Notes

1. Names that collide with Achronix reserved module names are prohibited.

Restore Implementation Dialog

The Restore Implementation dialog restores the database state of the active implementation from an Acxdb (`.acxdb`) archive file. After indicating the path to the archive file from which to restore the implementation, click **Finish** to restore the active implementation.

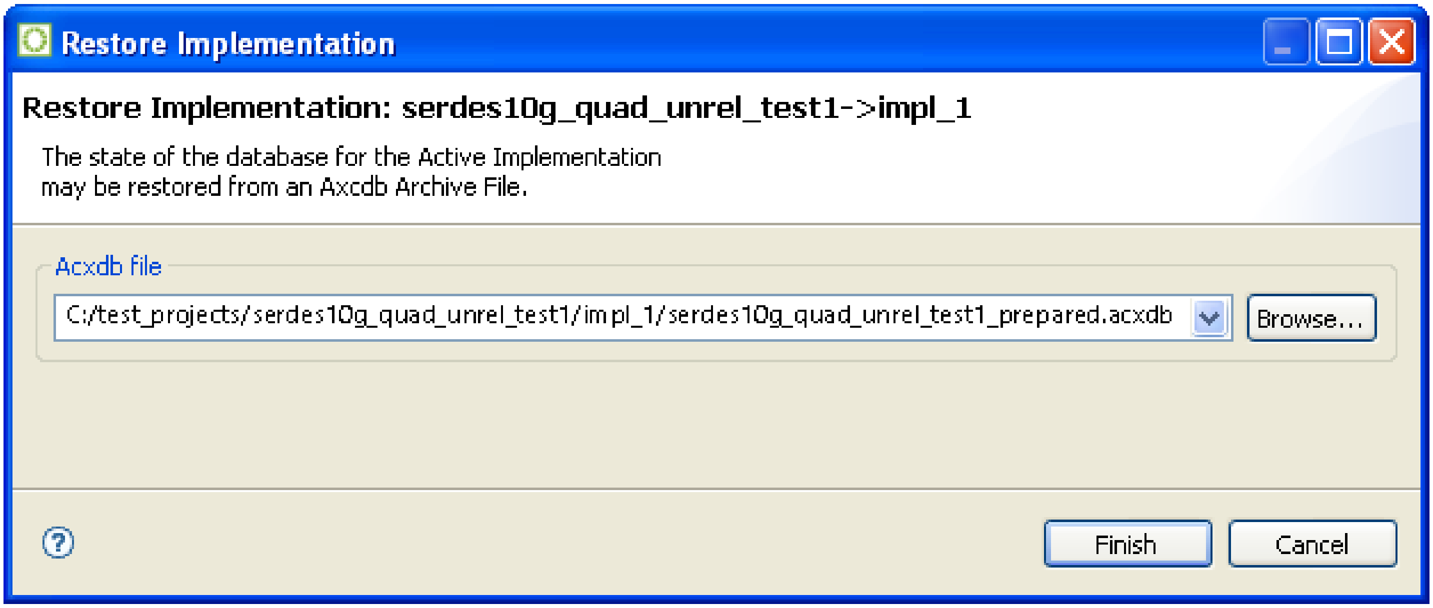


Figure 73: *Restore Implementation Dialog Example*

Table 100: *Restore Implementation Dialog Fields*

Field	Default	Description
Acxdb File	The newest `.acxdb` file in the directory	The path to the Acxdb archive file from which to restore the implementation. A drop-down list provides easy access to all other archive files in the directory.

Save Changed Properties Dialog

The Save Changed Properties Dialog allows saving to an `.sdc` file any properties that have been changed since the **Run Prepare** flow step was last executed.

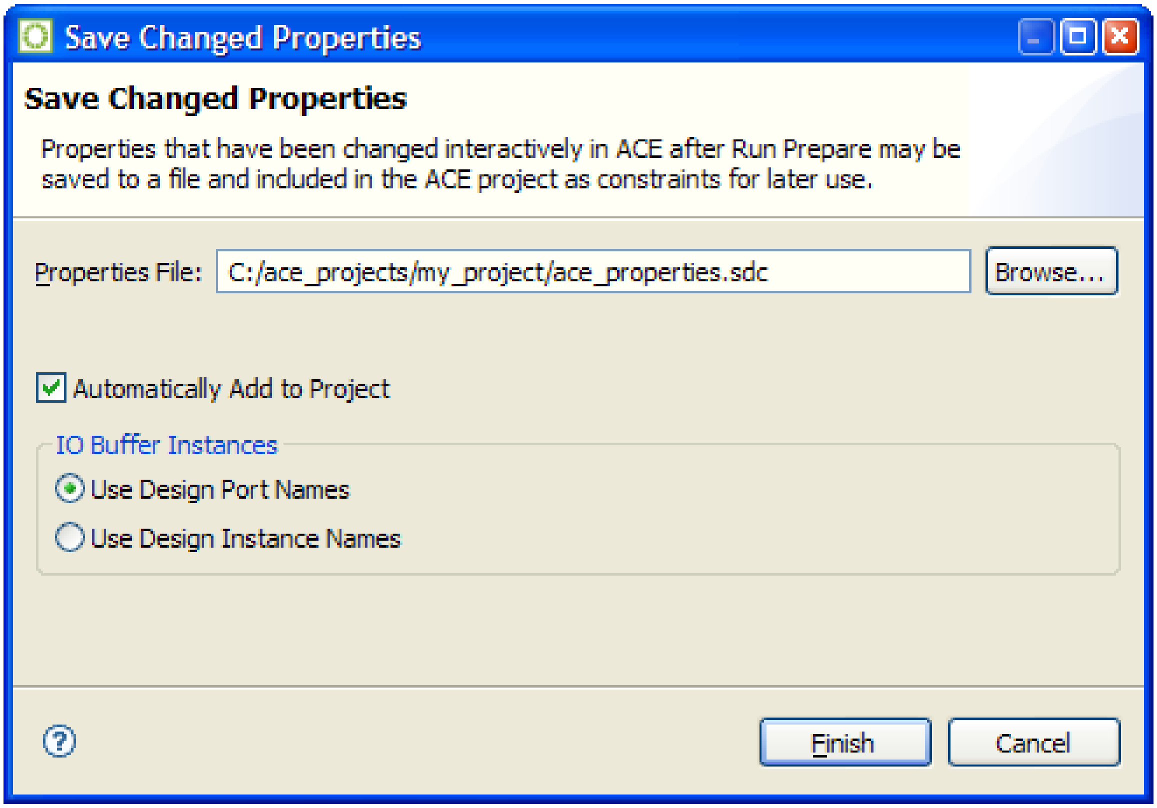


Figure 74: ***Save Changed Properties Dialog Example***

To perform this action without using the dialog, use the Tcl command `save_properties`.

See also: IO Assignment View (see page 74) and Managing I/Os (see page 431).

Save Implementation Dialog

The Save Implementation dialog saves the database state of the active implementation to an archive file (`.acxdb`). After indicating the path to the `.acxdb` file to which the implementation is saved and whether to include the log file, click **Finish** to save the active implementation.

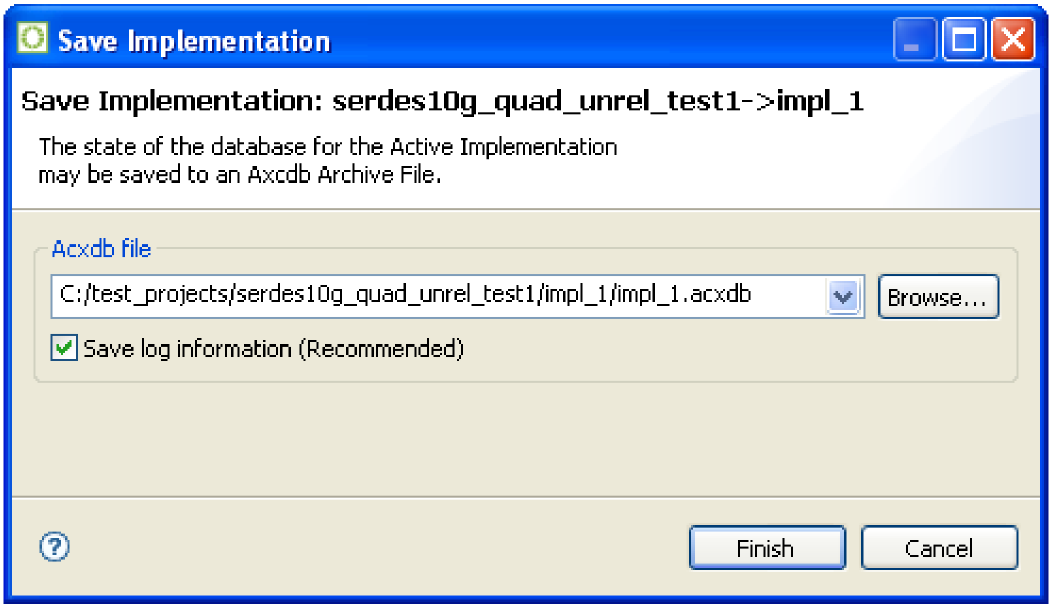


Figure 75: ***Save Implementation Dialog Example***

> **Note**
>
> Implementations may only be saved after the Run Prepare flow step has been completed. Prior to that, there is no meaningful content in the database to save.

Table 101: ***Save Implementation Dialog Fields***

Field	Default	Description
Acxdb File	`.acxdb`	The `.acxdb` archive file path to which the implementation is saved.
Save Log Information	On	If this field is checked, the log file for the current active implementation is included in the archive file.

Save Placement Dialog

The Save Placement Dialog saves the current placement to pre-placement constraints file(s). After selecting the appropriate options, click **Finish** to save the placement. This dialog can be triggered from the Floorplanner View (see page 53), the Search View (see page 132), and the Selection View (see page 136).

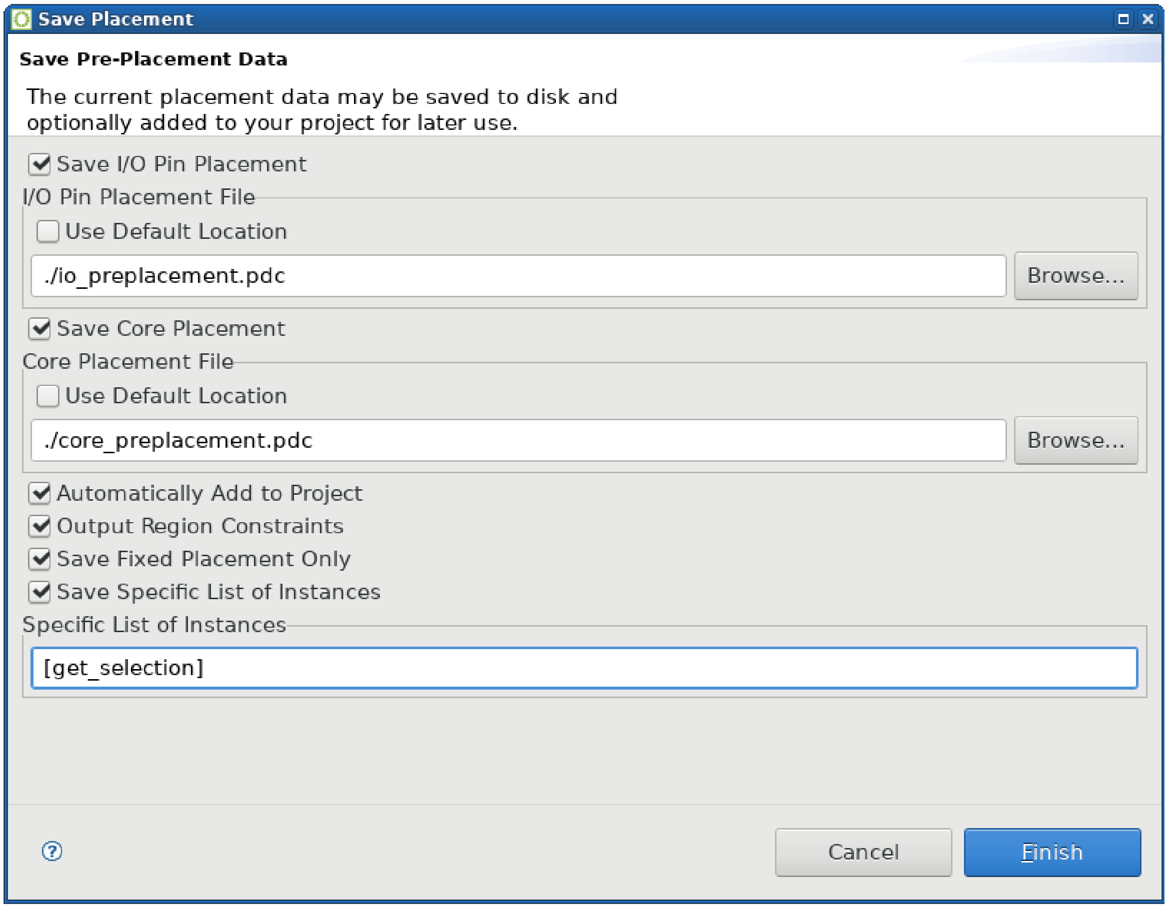


Figure 76: ***Save Placement Dialog Example***

Table 102: ***Save Placement Dialog Fields***

Field	Default	Description
Save I/O Pin Placement	Enabled	Indicates whether placement of instances in the Boundary ring should be saved.
I/O Pin Placement File		
Use Default Location	Enabled	Specifies using the default I/O pin placement file path.
(Text Box)	(blank)	Specify the I/O pin placement file path if the above option is disabled.
Save Core Placement	Enabled	Indicates whether placement of instances in the core fabric should be saved.

Field	Default	Description
Core Placement File		
Use Default Location	Enabled	Specifies using the default core placement file path.
(Text Box)	(blank)	Specify the core placement file path if the above option is disabled.
Automatically Add to Project	Enabled	Controls whether the output placement files are automatically added to the current project as pre-placement constraints files.
Output Region Constraints	Enabled	Controls whether Placement Regions and Placement Region Constraints (see page 354) are exported to the PDC file.
Save Fixed Placement Only[1]	Enabled	Controls whether only the current fixed-placement constraints are saved to the output files. If set, all other placement information (such as that generated by the placer) is ignored.
Save Specific List of Instances	Contextual	Enabled by default when created from the Search View (see page 132) or Selection View (see page 136). Disabled by default when created from the Floorplanner View (see page 53).
Specific List of Instances		
(Text Box)	(blank)	Enter a Tcl list of instance names, or a Tcl statement that returns a list of instance names. This list is then used (instead of all instances in the design) in combination with the other dialog settings when choosing what to save.

Table Notes

1. It is recommended to always use this option when saving pre-placement constraints.

Save Placement Regions Dialog

The Save Placement Regions Dialog appears after selecting the **Save Placement Regions** action in the Placement Regions view. This dialog allows saving placement region definitions, including the instance constraints for those placement regions.

See also: Saving Placement Region Constraints (see page 360).

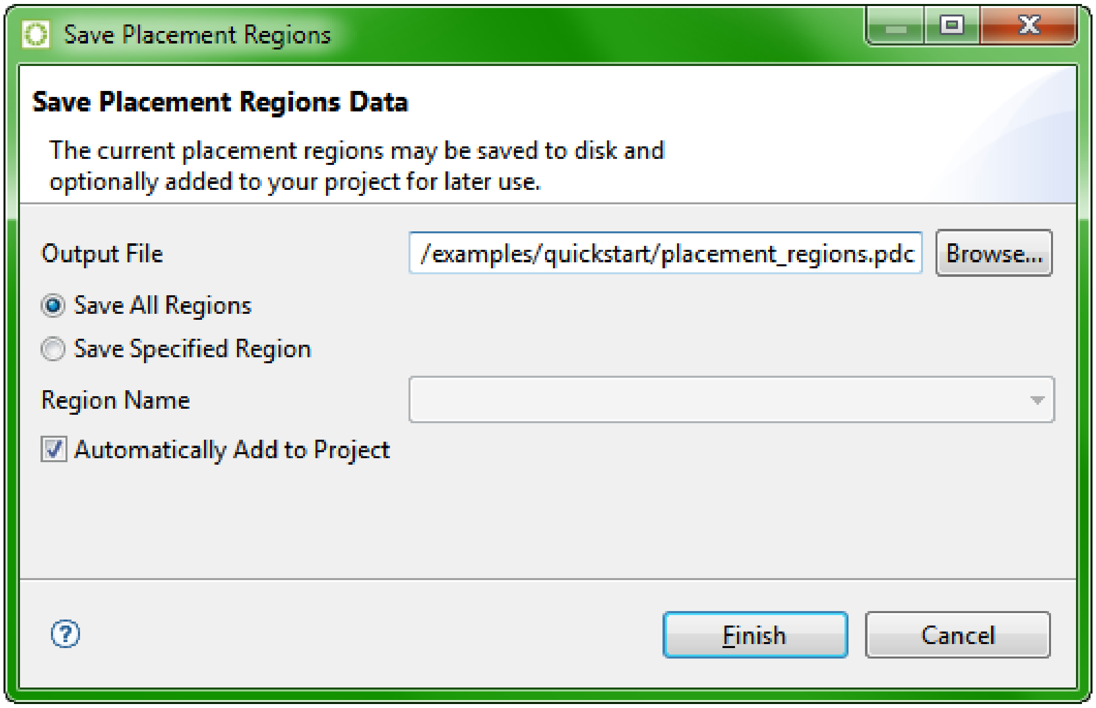


Figure 77: ***Save Placement Regions Dialog Example***

Table 103: ***Save Placement Regions Dialog***

Option	Description
Output File	The full path to the `.pdc` file which contains the region definitions and constraints.
Save all Regions	Saves the data for all regions listed in the Placement Regions view.
Save Specified Region	Saves the data for a single region (specified below).
Region Name	The name of the Placement Region to be saved. This field is disabled unless **Save Specified Region** is selected.
Automatically Add to Project	When selected, if the **Output File** is not already a member of the constraints for the active project, the file is added to the project as a constraint file.

Save Script File As Dialog

The Save Script File As dialog appears when the **Write Script for Schematic View** button ( ) is selected in the Critical Paths View (see page 48).

The Save Script File As dialog is used to create a Tcl script of `find` commands for the current list of critical paths. The script is intended for use in the schematic viewer of the synthesis tool.

After indicating a filename and location for the Tcl script, click **OK** to write the script to disk, or **Cancel** to close the dialog without saving.

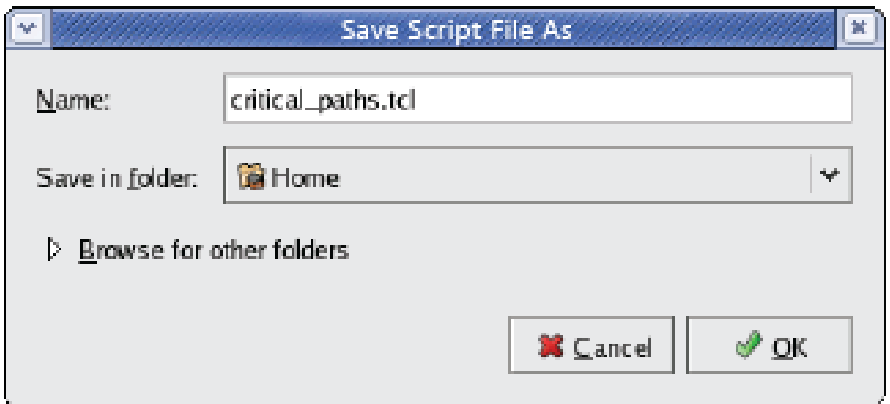


Figure 78: ***Save Script File As Dialog Example***

Search Filter Builder Dialog

The Search Filter Builder Dialog allows building simple or compound search filters for the Search View (see page 132) (the dialog is accessed from the **...** button in the **Filters** row of the Search View).

Search filters are used to find objects in the design based upon properties other than object name. Simple filters may be combined into a compound filter by joining them with Boolean operators. See Filter Properties (see page 244) for a table of the available filter properties with descriptions.

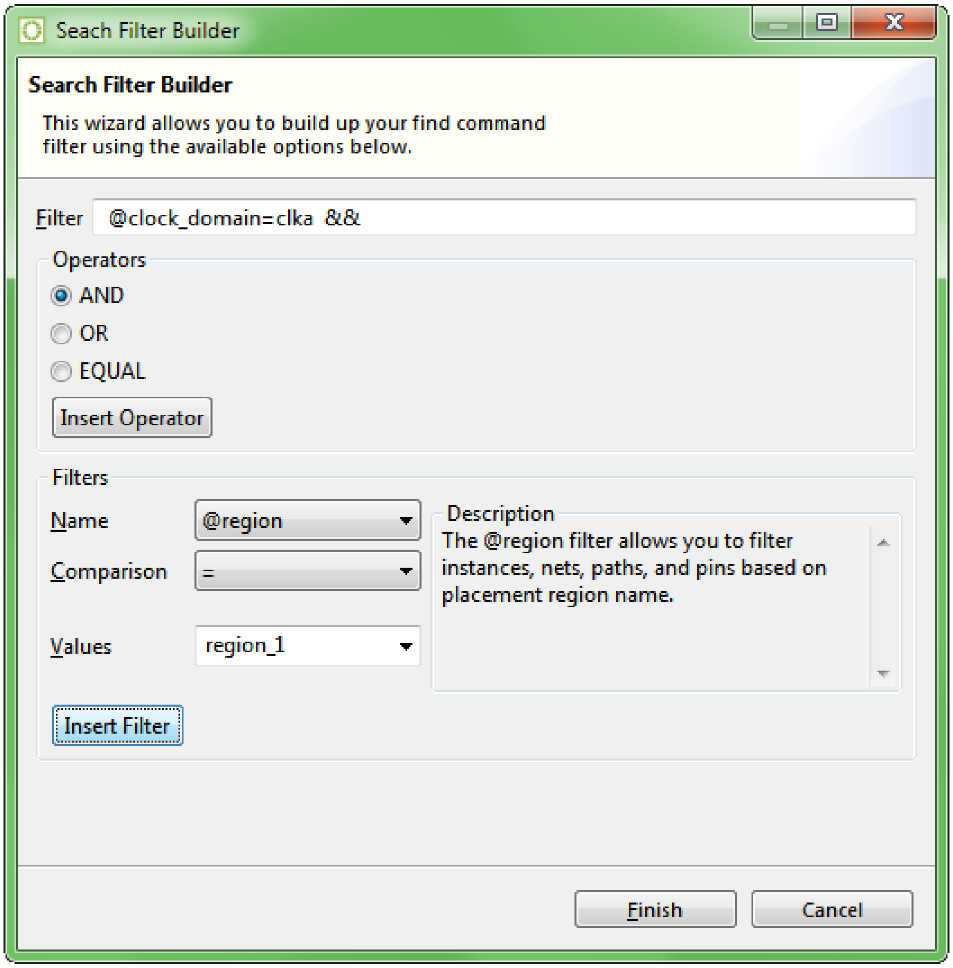


Figure 79: ***Search Filter Builder Dialog Example***

Table 104: ***Search Filter Builder Dialog Options***

Option	Description
Filter	The filter string itself. Type directly into this field, or populate this field by using the **Insert Operator** and **Insert Filter** buttons.
Operators	
AND	Select this radio button to join two filters into a compound filter where both sub-filters are true.
OR	Select this radio button to join two filters into a compound filter where either sub-filter is true.
EQUAL	Select this radio button to join two filters into a compound filter where the Boolean value of both sub-filters is identical.
Insert Operator	Click this button to insert the selected Boolean operator into the **Filter** field at the current cursor position within that field.

Option	Description
Filters	
Name	A combo box showing all choices of supported filter parameter names. The value of this field affects the content of the **Description** areas, as well as the possible values for the **Comparison**, and **Values** options. For a list of all available types of filters, see Filter Properties (see page 244).
Description	Provides a textual description of the current filter parameter selected in the **Name** field. This field may also provide hints or details on how the **Comparison** or **Values** fields may be populated.
Comparison	Selects from the set of possible comparisons relevant to the selected filter parameter **Name**. The contents of this combo box change according to the current **Name** selection.
Values	Enter the desired value(s) to compare against. For some filter parameter **Name** values, the combo box may contain possible values. The contents of this combo box change according to the current **Name** selection.
Insert Filter	Click this button to insert the selected filter specification into the **Filter** field at the current cursor position within that field.
Finish	Click this button to close the dialog, copying the current value of the **Filter** text field into the **Filters** field of the Search View.
Cancel	Click this button to close the dialog without changing the current value of the **Filters** field of the Search View.

Generate I/O Ring Design Files Dialog

The Generate I/O Ring Design Files Dialog selects the directory into which all the customized I/O Ring design files are output, including:

- Complete package ball pin assignment, power, and utilization reports
- Pin placements PDC, Verilog wrappers, and port lists for the Core user design
- The full I/O Ring bitstream, which is automatically combined with the Core user design bitstream in ACE at the end of the normal place-and-route flow for the Core user design
- Customized I/O Ring simulation files, including Verilog wrappers for the top-level and I/O Ring configuration data

The files generated are based upon the I/O Ring IP Configuration files (`.acxip`) in the active ACE project created via the active IP Configuration Editor (see page 26). See also: Creating an IP Configuration (see page 306) and I/O Designer Toolkit Views (see page 66).

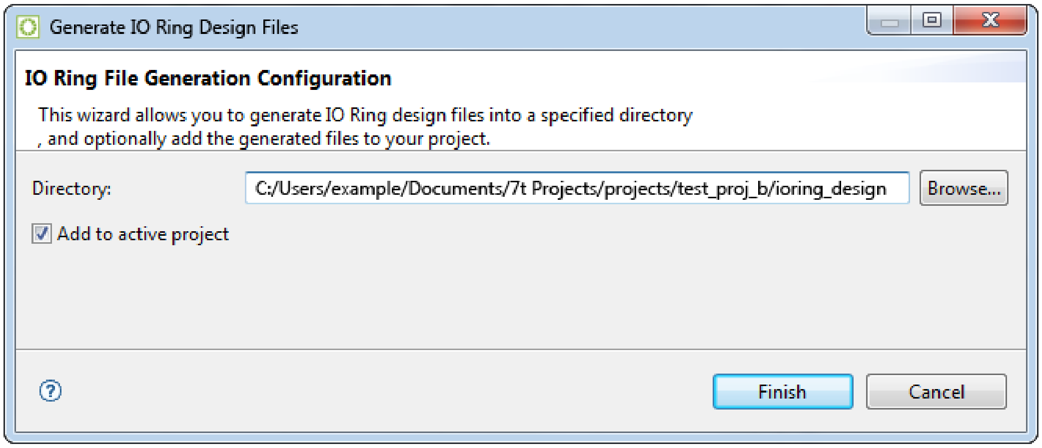


Figure 80: *Generate I/O Ring Design Files Dialog Example*

Table 105: *Generate I/O Ring Design Files Dialog Fields*

Field	Default	Description
Directory	<active_ace_project_dir>/ioring_design	Selects the target directory path for the I/O ring design files when generated.
Add to active project	Selected	When selected, the necessary generated I/O ring design files (e.g., PDC pin placement constraints, Verilog wrappers, and SDC files for the core user design logic) are automatically added to the active ACE project file.

Create a SecureShare Zip File Dialog

The Create a SecureShare Zip File Dialog allows choosing or refining the contents of a SecureShare file, which is then zipped (and placed in the chosen directory), ready for delivery to Achronix Technical Support at https://support.achronix.com/hc/en-us/. By default, the Zip file contains all of the important details of the design, enabling Achronix support engineers to examine the design and all output files created during the ACE flow.

Optionally, files may be removed from the default lists of chosen files if it is preferred that those files are not sent to Achronix. In addition, the information may be optionally encrypted in the SecureShare Zip file.

There are presently three main sections of information in the dialog:

1. The Configuration section
2. The ACE input files section
3. The ACE output files section

More sections are planned to appear in future ACE releases, covering additional support categories, such as synthesis.

This dialog is accessed by selecting **Help** → **Start SecureShare**, or by using the keyboard shortcut **Ctrl-Alt-Shift-s**. When the dialog is shown, it is completely populated by default based upon information gathered from the Active Project and Implementation (see page 221). See also Using the ACE SecureShare Tool to Create a Support Zip File (see page 435).

Configuration

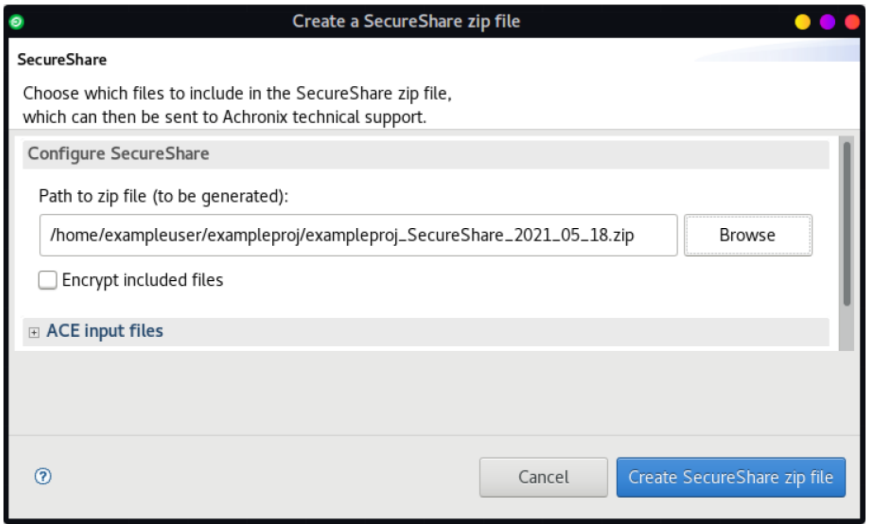


Figure 81: *Create a SecureShare Zip File Dialog*

Table 106: *Create a SecureShare Zip File Options*

Control	Description
Path to Zip file (to be generated)	This is the SecureShare file to be generated by ACE when the **Finish** button is clicked. By default, the Zip file is written to the output subdirectory of the active implementation, with the file name being the design name as a prefix, followed by "SecureShare", then a suffix comprised of the date the file is generated. For example, `<active_impl_output_directory>/<active_project_name>_SecureShare_<yyyy>_<mm>_<dd>.zip`.
Encrypt included files	Defaults to false/unchecked. When checked, the Zip file (above) is encrypted, and placed in an additional file with the extension `.zip.encrypted`.

ACE Input Files

Each file category (other than the project file) can hold an arbitrary number of files. Files can be added to (or removed from) any of these categories as desired. Clicking any **Add** button pops up a file selection dialog (which defaults to the correct file extensions for filtering) where one or more files to be added to the associated file list can be chosen.

> **Note**
>
> Be aware that every file selection dialog also allows setting the file extension filter to "*.*", allowing any filename to be chosen.

Selecting one or more files from a file list category enables the associated **Remove** button which, when clicked, removes the selected files from the list.

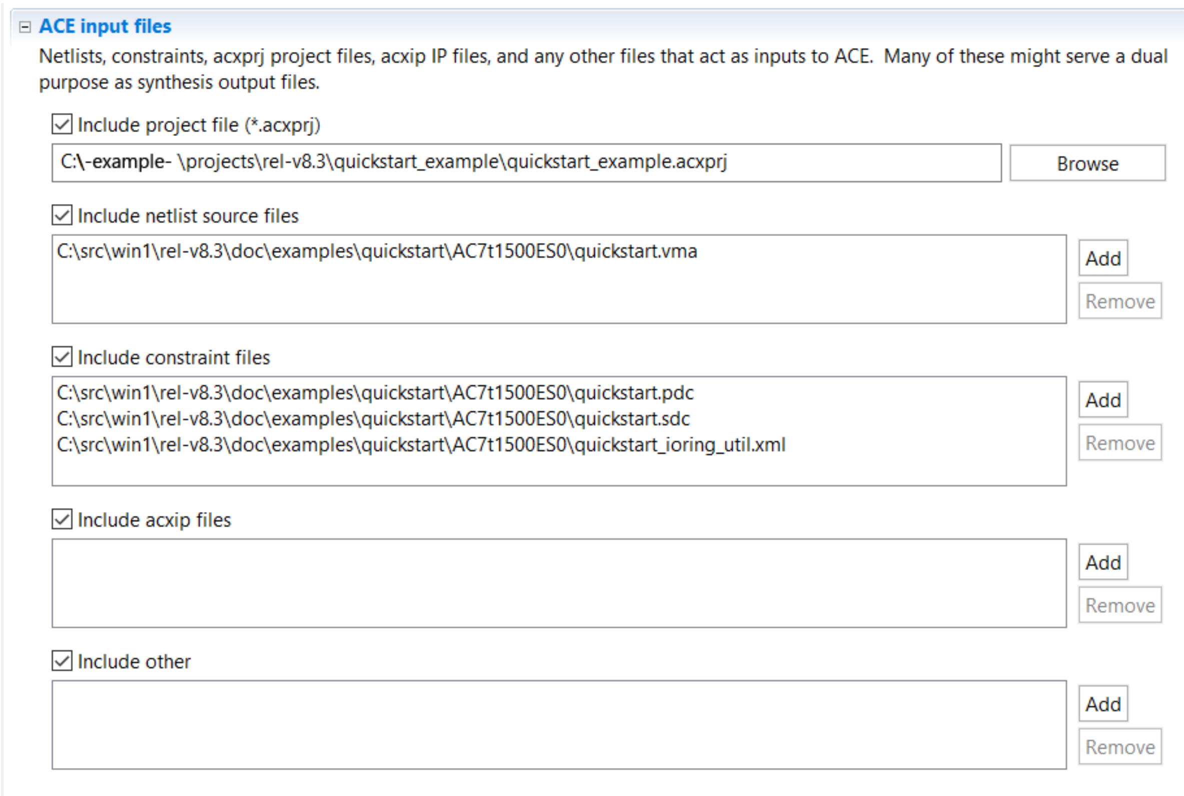


Figure 82: ***ACE Input Files Dialog***

Table 107: ***SecureShare Input File Categories***

File Category	Description
Project file (*.acxprj)	Defaults to the `.acxprj` file for the active project.
Netlist source files	Defaults to all the netlist source files (`*.v`, `*.vm`, `*.vma`, `*.sv`, `*.vhdl`) found within the active project.
Constraint files (1)	Defaults to all the enabled constraint files (`*.hex`, `*.pdc`, `*.prt`, `*.scf`, `*.sdc`, `*.xml`) for the active implementation within the active project.
`*.acxip` files	Defaults to all of the `.acxip` files found within the active project.
Other	Defaults to empty. This entry allows the addition of any other arbitrary ACE input files which are not covered by the earlier categories.

Table Notes

1. Constraint files within the active project which are not part of the active implementation are not included by default.

ACE Output Files

Similar to input files, each output file category (other than the project file) can hold an arbitrary number of files. Add files to (or remove files from) any of these categories as desired. Clicking any **Add** button pops up a file selection dialog (which defaults to the correct file extensions for filtering) where one or more files to be added to the associated file list can be chosen.

> **Note**
>
> Be aware that every file selection dialog also allows setting the file extension filter to "*.*", allowing any filename to be chosen.

Selecting one or more files from a file list category enables the associated **Remove** button which, when clicked, removes the selected files from the list.

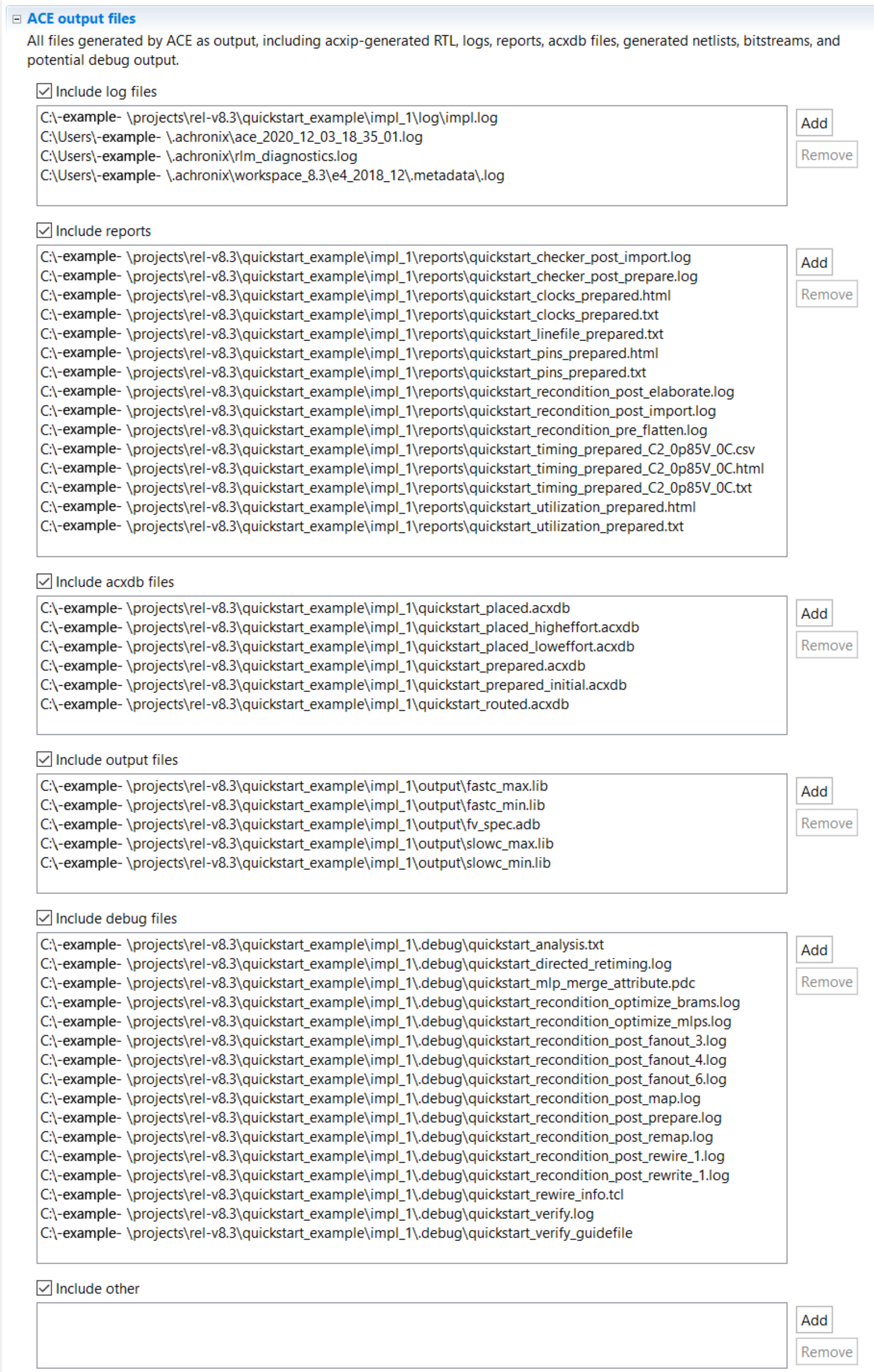


Figure 83: ***ACE Output Files Dialog***

Table 108: *SecureShare Input File Categories*

File category	Description
Log files	Defaults to a list of all the files found within the active implementation log subdirectory (which might include several multiprocess logs), plus: • Any detected GUI log files • The latest ACE session log file • The latest ACE `rlm_diagnostics.log` file (to help track down any ACE license-related problems).
Reports	Defaults to a list of all the files found within the active implementation reports subdirectory, plus the latest Multiprocess Summary Report for the active project.
`*.acxdb` files	Defaults to a list of all of the `.acxdb` files found within the active implementation directory.
Output files	Defaults to a list of all the files found within the active implementation output subdirectory.
Debug files	Defaults to a list of all the files found within the active implementation `.debug` subdirectory.
Other	Defaults to empty. This entry allows adding any other arbitrary ACE-related output files which are not covered by the earlier categories.

Load Acxdb Dialog

After cancelling the Flow (see page 221) in the Flow View (see page 61), the Load Acxdb dialog allows restoring the database of the active implementation to a previously saved state (the state found in the `.acxdb` archive file). This restoration is useful because canceling a running flow might otherwise leave the database in a partially processed state.

After populating the **Acxdb File** field with the desired archive file (which defaults to the `.acxdb` file with the most recent timestamp), click **Finish** to restore the implementation to the state preserved in the file. To avoid loading a file and to proceed with the database in the current (incomplete flow) state, click **Cancel**.

Note

When canceling the flow, this dialog only appears if the run is cancelled after the **Run Prepare** stage is complete.

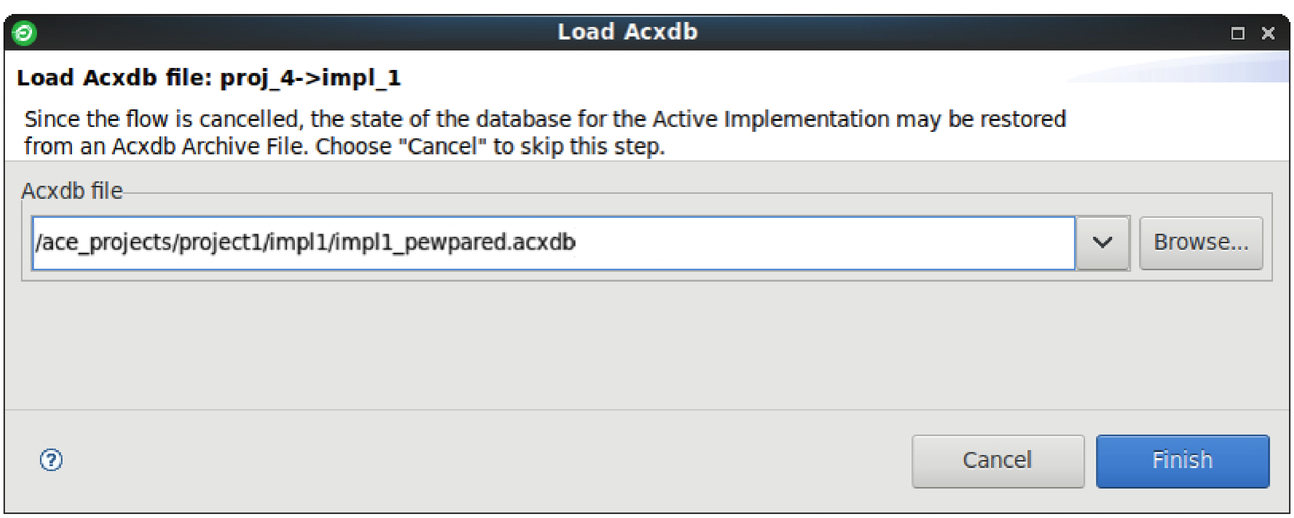


Figure 84: *Load Acxdb Dialog Example*

Table 109: ***Load Acxdb Dialog Fields***

Field	Default	Description
Acxdb File	The latest archive file file found in the active implementation directory.	The file path to the desired `.acxdb` archive file, which is used to restore the implementation to an earlier saved state. Enter the desired path directly into this field. A drop-down list provides easy access to all archive files in the default directory for the active implementation. The **Browse...** button allows graphically navigating to alternate files.

Plot Serdes Diagram Dialog

The Plot Serdes Diagram Dialog appears after selecting **Plot Serdes Diagram** from the option menu option when right-clicking a Serdes Lane in the I/O Layout Diagram View (see page 72). The dialog allows plotting a diagram (Eye plot, Histogram, or Bathtub) for the selected Serdes RX lane using the built-in capture hardware.

> **Note**
>
> This tool requires the free Matlab runtime executable to be installed in addition to ACE with an open JTAG connection in your ACE session to access the Serdes HW on the chip.

See also: Plotting Serdes Rx Diagrams using JTAG (see page 439).

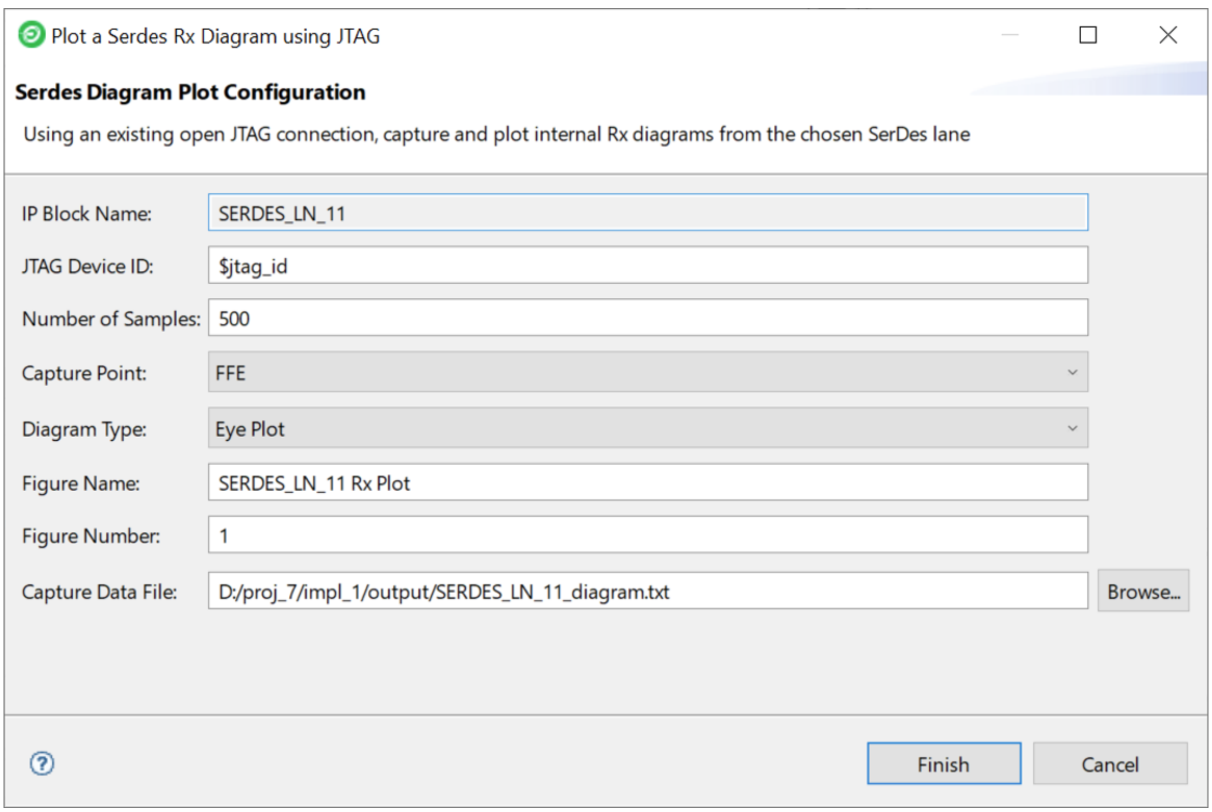


Figure 85: ***Plot Serdes Diagram Dialog Example***

Table 110: *Plot Serdes Diagram Dialog Fields*

Option	Description
IP Block Name	The name of the Serdes IP Block selected from the right-click option menu in the I/O Layout Diagram View.
JTAG Device ID	The ID of the connected JTAG device (an FTDI USB chip or a Bitporter2 pod). This ID can be specified with a hard-coded string such as "AC12345" or with a Tcl variable.
Number of Samples	The number of samples to capture in the Serdes Rx capture hardware. The default for an Eye Plot is 500 samples. Over JTAG, this requires approximately 10 minutes to read back and plot. More samples produces a more accurate diagram, while less samples can reduce runtime.
Capture Point	Selects the internal Serdes block from which to capture data.
Diagram Type	Select an Eye Plot, Histogram, or Bathtub plot.
Figure Name	Specifies a label to appear on the generated diagram image.
Figure Number	Specifies a figure number label to appear on the generated diagram image.
Capture Data File	Enter the file path to the diagram capture data retrieved from the Serdes hardware via JTAG. Click **Browse...** to graphically select the file. This file can be used to plot multiple diagram types from the same data.

Toolbars

There are three kinds of toolbars in the Workbench (see page 24): main, view, and trim.

The main toolbar, sometimes called the Workbench toolbar, is displayed at the top of the Workbench window directly beneath the menu bar. The contents of this toolbar change based on the active perspective. Items in the toolbar might be enabled or disabled based on the state of either the active view (see page 31) or editor (see page 26). Sections of the main toolbar can be rearranged using the mouse.

There are also individual view toolbars, which appear in the title bar of a view (see page 31). Actions in a view toolbar apply only to the view in which they appear. Some view toolbars include a Menu button, shown as an inverted triangle, which contains additional actions for that view.

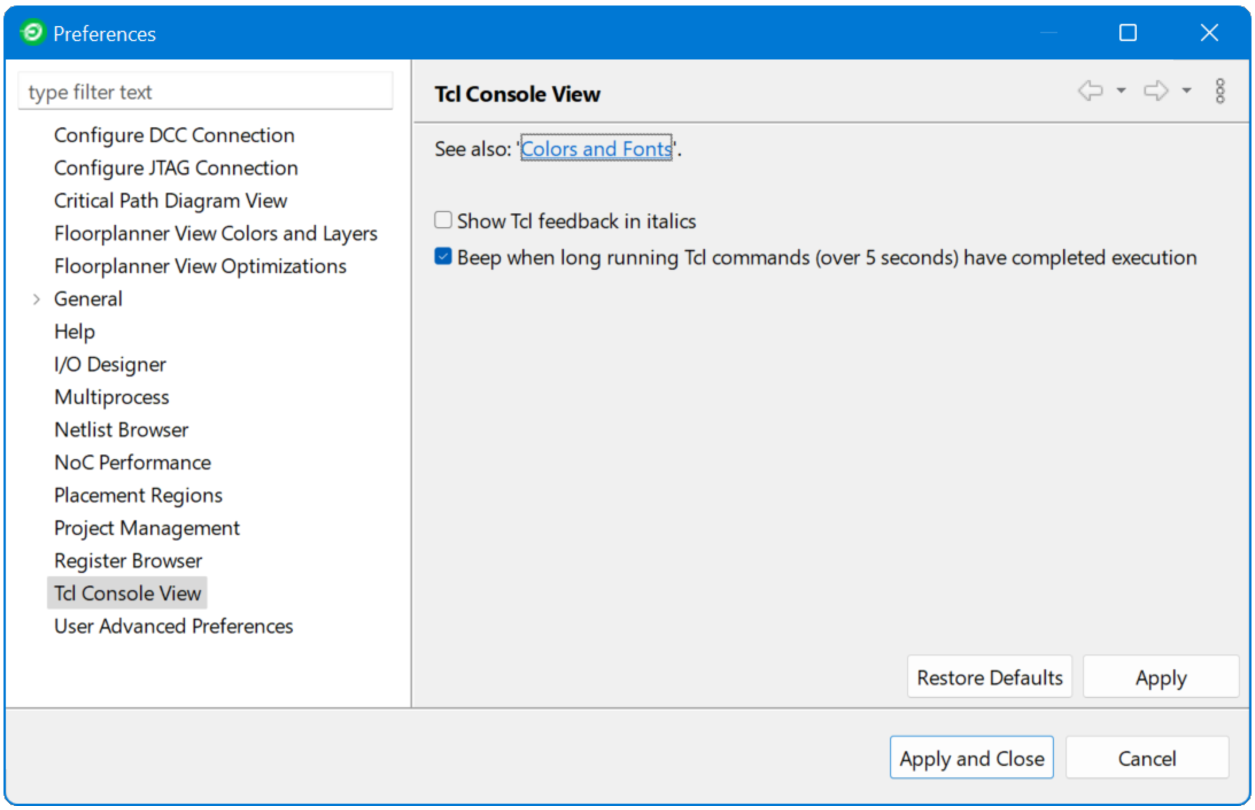
Minimizing a view/editor tab stack also produces a toolbar in the trim at the outer edge of the workbench window (a Trim Stack). This bar contains an icon for each of the views in the stack and/or a single icon for each stack of editors. Clicking one of these icons results in the view/editor being displayed as an overlay onto the workbench window.

In all cases, when the cursor is positioned over a toolbar button, a tooltip describing its function appears.

Preferences

The Preferences dialog sets user preferences. The dialog pages can be searched using the filter function. To filter by matching the page title, simply type the name of the page being sought, and the available pages are presented below. The filter also searches on keywords such as "appearance" and "text". The history controls allow navigation through previously viewed pages. To step back or forward several pages at a time, click the drop-down arrow to see a list of the most recently viewed preference pages.

The Preferences dialog can be found from the main workbench **Window** menu under **Window** → **Preferences…**.



Configure DCC Connection Preference Page

The Configure DCC Connection Preference Page configures the Preferences (see page 185) for the DCC (Demo Command and Control) connection, as used with the HW Demo View (see page 64).

See also: Configuring the DCC Connection (see page 330), Running the HW Demo (see page 361).

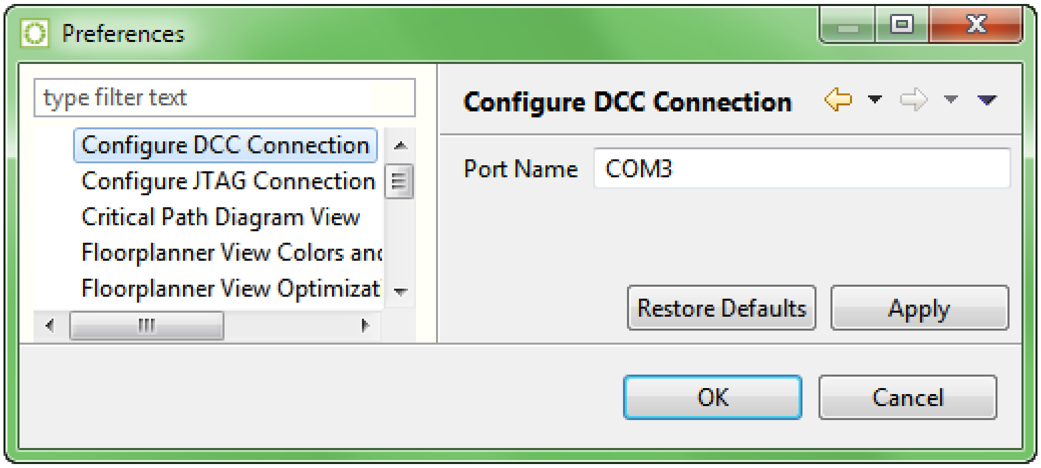


Figure 86: ***Configure DCC Connection Preference Page Example***

Table 111: ***Configure DCC Connection Preference Page Options***

Option	Description
Port Name	Enter the serial port name used for the DCC connection. For further information about determining which serial port should be used, please see Configuring the DCC Connection (see page 330).

Configure JTAG Connection Preference Page

The Configure JTAG Connection Preference Page configures which Bitporter or FTDI FT2232H device is used to communicate via JTAG to the desired Device Under Test (the test board). The page also specifies where the Achronix device is found in the JTAG scan chain, which might potentially contain multiple Achronix and non-Achronix devices.

> **Warning!**
>
> Bitporter JTAG pods can be damaged if connected improperly. Before attempting to use a Bitporter JTAG pod, please consult the *JTAG Configuration User Guide* (UG004) at https://www.achronix.com/documentation/jtag-configuration-user-guide-ug004.

Multiple views use these configuration preferences, including the Download View (see page 51), the Snapshot Debugger View (see page 139), and the HW Demo View (see page 64). Specialized functionality for some IP Configuration Editors might also use these JTAG preferences.

The section Configuring the JTAG Connection (see page 331) explains the proper use of all fields of this page in detail.

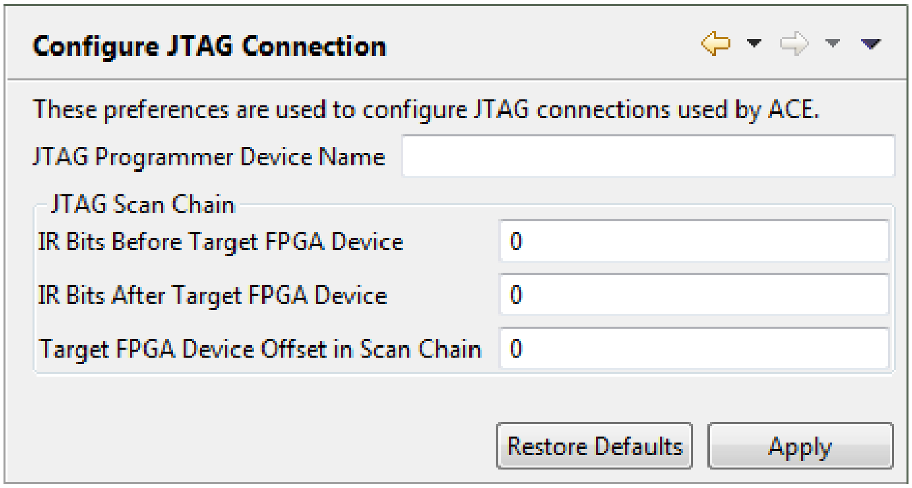


Figure 87: ***Configure JTAG Connection Preference Page Example***

Table 112: ***Configure JTAG Interface Preference Page Options***

Option	Description
JTAG Programmer Device Name	The name of the JTAG programmer device which should be used for all ACE JTAG interactions with the chosen FPGA. If the name is not specified, auto-detection of JTAG programming devices is attempted. (1) Specifying the JTAG device by name can save several seconds of initialization time on every JTAG connection, even if only one pod is connected.
JTAG Scan Chain	
IR Bits Before the Target FPGA Device	Sets the (decimal) number of instruction register bits between the board JTAG TDI pin and the target device. (2)
IR Bits After the Target FPGA Device	Sets the (decimal) number of instruction register bits between the target device and the board JTAG TDO pin. (2)
Target FPGA Device Offset in Scan Chain	Sets the device count (in decimal) between the board JTAG TDI pin and target FPGA device. (2)

Table Notes

1. Auto-detection can only be used safely when just one pod/device is connected. If more than one pod/device is automatically detected, pod interactions fail, stating that it is required to specify which pod/device to use.
2. The default value of zero is always correct for single-device JTAG scan chains.

Multi-device JTAG Scan Chain (IEEE 1149.1) Example

The following high-level diagram summarizes how ACE must be configured for JTAG daisy-chains. For an explanation of daisy-chained JTAG scan chains, visit https://en.wikipedia.org/wiki/Jtag#Daisy-chained_JTAG_.28IEEE_1149.1.29.

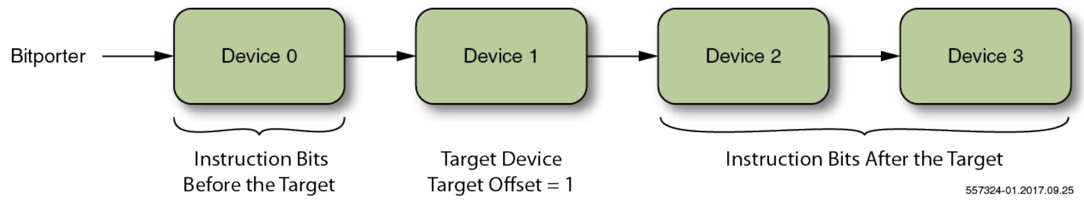


Figure 88: ***Example High-Level Diagram of a Multi-device JTAG Scan Chain***

When multiple FPGA devices are attached to the same JTAG scan chain, the target FPGA device must be specified. The FPGA device closest to the Bitporter (more accurately, closest to the board JTAG TDI pin) has a target offset of 0.

Because different FPGA devices have different instruction register (IR) sizes, the total IR bit length before and after the target must be specified as well. Achronix devices have a JTAG instruction register size of 23 bits. Hence, in the above example diagram, if all devices were Achronix FPGA devices, there would be 23 IR bits before the target FPGA device (23 IR bits within the target FPGA device) and 46 IR bits after the target FPGA device.

> **Note**
>
> For a more thorough explanation regarding ACE multi-device JTAG scan chain configurations, refer to Configuring the JTAG Connection (see page 331).

Critical Path Diagram View Preference Page

The Critical Path Diagram View Preference Page configures the display Preferences (see page 185) of the Critical Path Diagram View (see page 45).

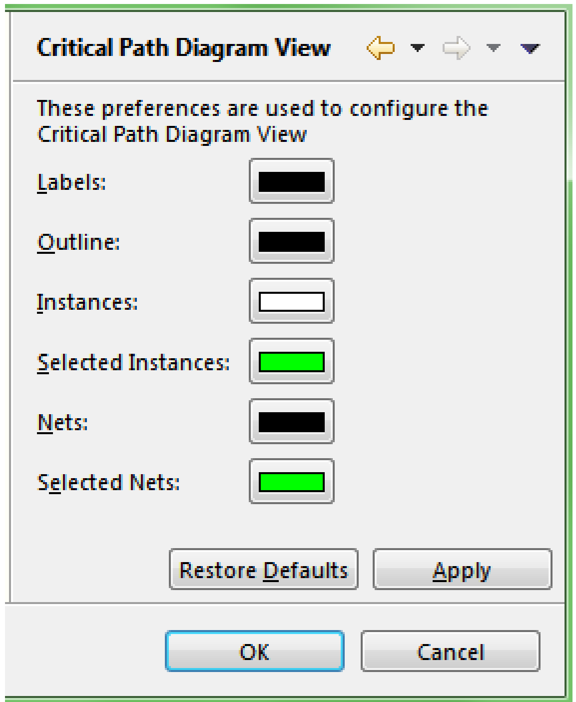


Figure 89: *Critical Path Diagram View Preference Page Example*

Table 113: *Critical Path Diagram View Preference Page Options*

Option	Description
Labels	Configures the color of the label text printed for graph nodes and arrows in the diagram.
Outline	Configures the color of the outline of the graph nodes in the diagram.
Instances	Configures the background color of graph nodes in the diagram.
Selected Instances	Configures the background color of graph nodes representing Instances in the ACE Selection Set in the diagram.
Nets	Configures the color of the arrows in the diagram.
Selected Nets	Configures the color of arrows representing Nets in the ACE Selection Set in the diagram.
Restore Defaults	Returns all preferences on this page to their ACE default values.
Apply	Immediately applies any configuration changes on this page to the current diagram in the Critical Path Diagram View (see page 45). These config values are also saved and are used in all future ACE sessions. The Preferences dialog stays open to allow other preference configuration changes, if desired.
OK	Immediately applies any preference configuration changes (including on other preference pages). These config values are also saved and are used in all future ACE sessions.

Option	Description
Cancel	Discards any preference configuration changes made since the dialog was opened (or since the last time the **Apply** button was clicked in the dialog, whichever was most recent).

Floorplanner View Colors and Layers Preference Page

The Floorplanner View Colors and Layer Preference Page configures the colors of multiple layers (and states within the layers) for the Floorplanner view (see page 53). Additionally, options are provided allowing a degree of control over the display priorities of the possible Instance States (see page 243) and Net/Route highlighting vs. selection. For more about Highlighting, see Highlighting Objects in the Floorplanner View (see page 315). For more about Selection, see the Selection View (see page 136).

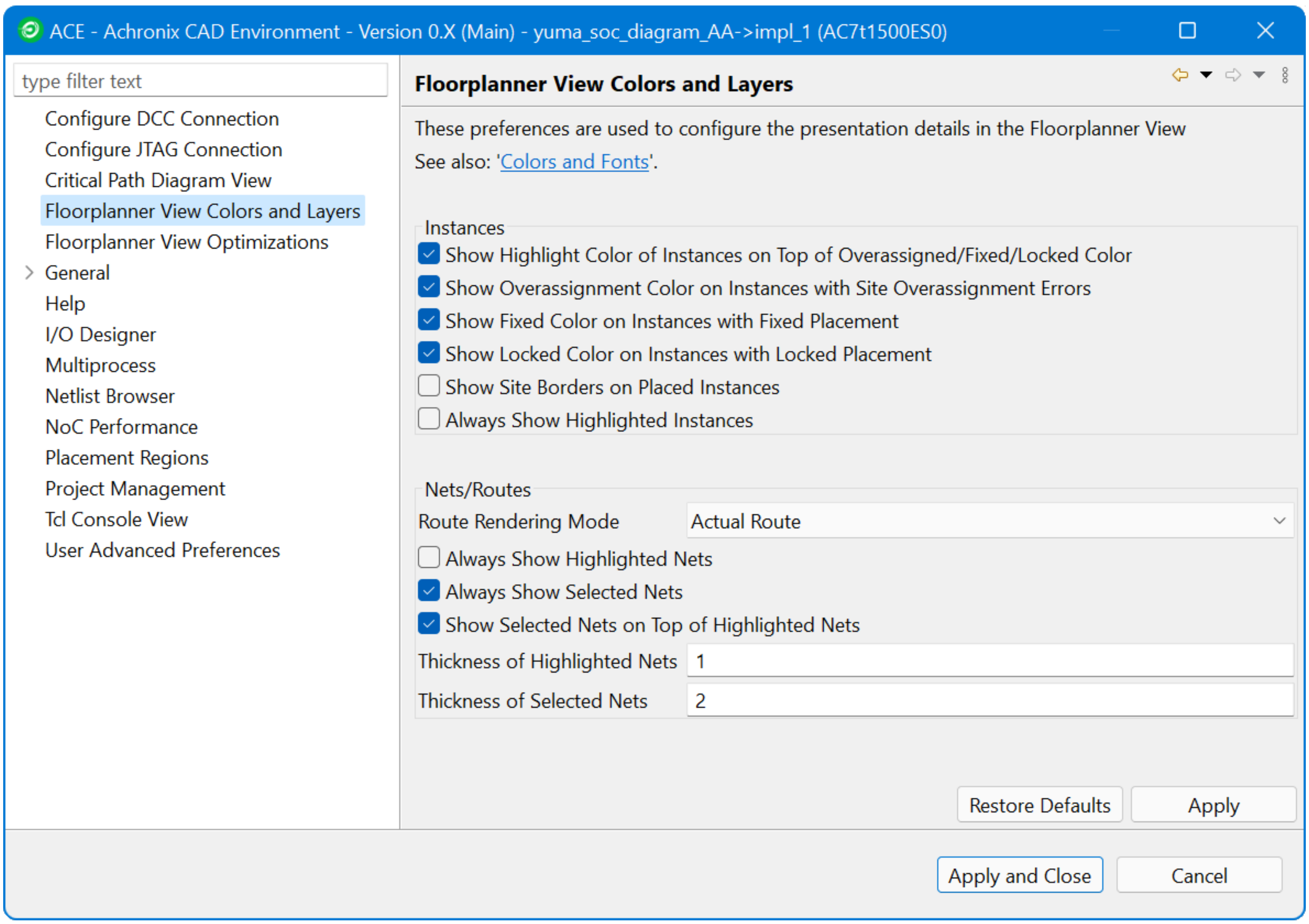
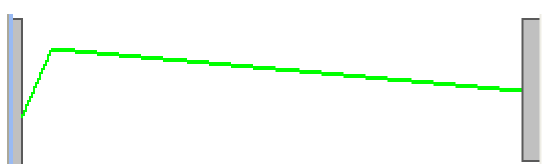
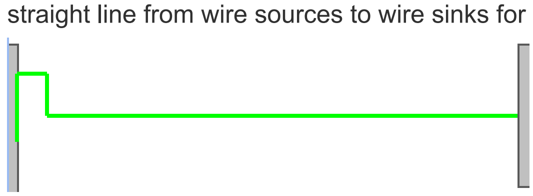


Figure 90: *Floorplanner View Colors and Layers Dialog*

The meanings of the various Instance States (see page 243) are defined elsewhere. While Instances can have multiple states at once, they can only show a single state at a time, thus there is some ability to alter the display priority of the various Instance states.

Similarly, the display of the Floorplanner Nets/Routes layers can be adjusted. (Both Clock and non-Clock nets are affected identically by these settings.)

Table 114: ***Instances and Nets/Routes Preferences***

Preference	Description
Instances	
Show Highlight Color of Instances on Top of Overassigned/Fixed /Locked Color	When enabled, the Instance Highlight color has a higher priority than all other Instance states except Selection.
Show Overassignment Color on Instances with Site Overassignment Errors	Allows toggling whether site over-assignment errors are displayed visually on the Floorplanner view.
Show Fixed Color on Instances with Fixed Placement	If disabled, both fixed placement and non-fixed placement instances are shown in the same color, grey by default.
Show Locked Color on Instances with Locked Placement	If disabled, locked and non-locked instances are shown with the same color.
Show Site Borders on Placed Instances(1)	If enabled, all placed instances are rendered with the Site border color as an outline around the instance. If disabled, placed instances are rendered without a site border.
Always Show Highlighted Instances	If enabled, Highlighted instances are always displayed, even if their layer (Instances/Selected Instances) is otherwise disabled. If disabled, Highlighted instances are only displayed when their layer (**Instances/Selected Instances**) is enabled.
Nets/Routes	
Route Rendering Mode	Allows altering how Non-clock Routes and Clock Routes are drawn (when the **Non-clock Routes** or **Clock Routes** layers are enabled, respectively, in the Floorplanner palette). Choices are **Actual Route** and **1 Corner**. •  **Actual Route** mode draws a single straight line from wire sources to wire sinks for each route segment. •  **1 Corner** mode draws two lines for each wire: from each wire source, draws a vertical line to the wire sink Y coordinate, then a horizontal line to the wire sink; no diagonal lines are used, which speeds rendering in complex designs.

Preference	Description
Always Show Highlighted Nets	If enabled, Highlighted nets are always displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Highlighted nets are only displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Always Show Selected Nets	If enabled, Selected nets are always displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Selected nets are only displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Show Selected Nets on Top of Highlighted Nets	Allows changing whether the Selection or Highlight color takes priority, and is painted "on top" of the other state during rendering.
Thickness of Highlighted Nets	The highlight color of a net is rendered this many pixels wide.
Thickness of Selected Nets	The selection color of a net is rendered this many pixels wide.

Table Notes

1. **Caution:** When **Show Site Borders on Placed Instances** is enabled while the view is extremely zoomed out, the site border render color might actually hide the placement state color. Thus, this preference is disabled by default.

Note

- The **1 Corner** route drawing mode is significantly faster (up to 5×) than the **Actual Route** drawing mode, but it can make individual routes harder to differentiate from each other. When Floorplanner performance is a concern, use **1 Corner** mode when possible, and only switch to **Actual Route** mode when needed.
- It is possible to show both the selection and highlight state of a single net simultaneously. Simply ensure the higher priority (top) state has a narrower thickness than the lower priority (bottom) state, and the bottom state color is rendered as a "halo" effect around the top state color.
- Having the "Always Show..." preferences enabled might be most useful in designs with massive numbers of nets/routes. Both the Clock and Non-clock route layers can be disabled, and then Select or Highlight the interesting routes, and only the Selected and/or Highlighted routes are displayed, without the distractions of the less-interesting routes cluttering up the view.

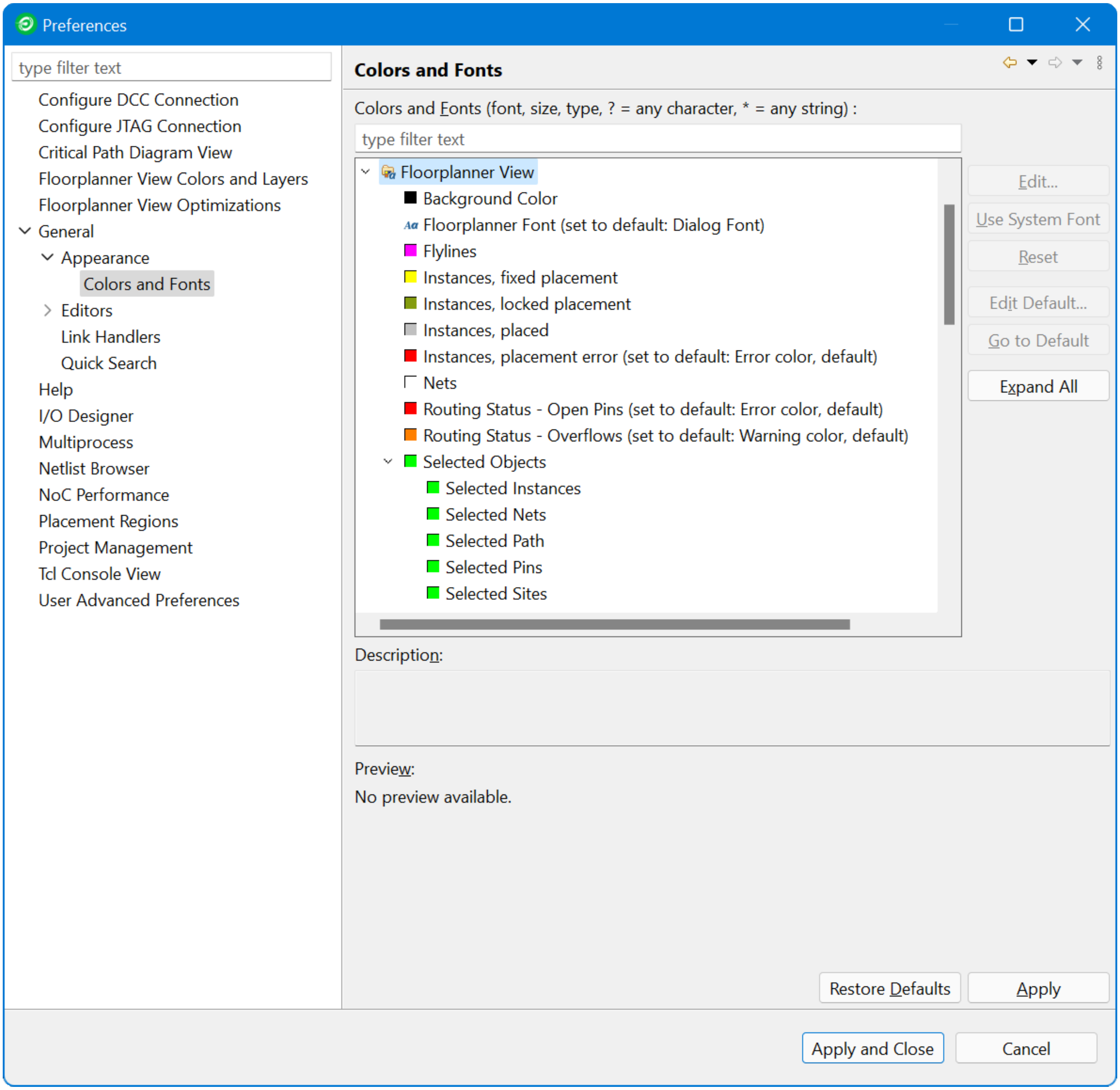


Figure 91: ***Colors and Fonts | Floorplanner View section***

Table 115: ***Color Preferences***

Color Preference	Description
Background Color	Used to render the background of the device.
Instances, placed	Represents instances with default (Soft) placement.
Instances, fixed placement	Represents instances with Fixed placement.
Instances, locked placement	Represents instances with Locked placement.
Instances, placement error	Represents an instance that shares a site assignment with another instance. Since a site can only legally contain a single instance, this is an error state.
Nets	Represents all net Routes for both clock and non-clock nets.
Flylines	Flylines are only rendered for Selected Instances, and only when the Layer called **Selected Instance Flylines** is enabled. These are straight single-segment lines directly connecting a net source instance and sink instance, where either the source or sink is a Selected Instance.
Selected Instances	Represents any Instance that is a member of the Selection Set (and is thus also visible in the Selection View (see page 136)).
Selected Nets	Represents any Net that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Pins	Represents any Pin that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Paths	Represents any Path that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Sites	Represents any Site that is a member of the Selection Set (and is thus also visible in the Selection View).
Routing Status – Open Pins	Represents Open Pins, the endpoints of Open Connections (which are the unrouted portions of a net). Open Pin squares are only visible when enabled in the Layers section of the Floorplanner view Palette.
Routing Status – Overflows	Represents Overflow pins, a rare routing error state. Overflow diamonds on pins are only visible when enabled in the Layers section of the Floorplanner view Palette.

Floorplanner View Optimizations Preference Page

The Floorplanner View Optimizations Preference Page configures rendering optimizations for the Floorplanner View (see page 53). The following Floorplanner View Optimations Page example includes default values for Windows:

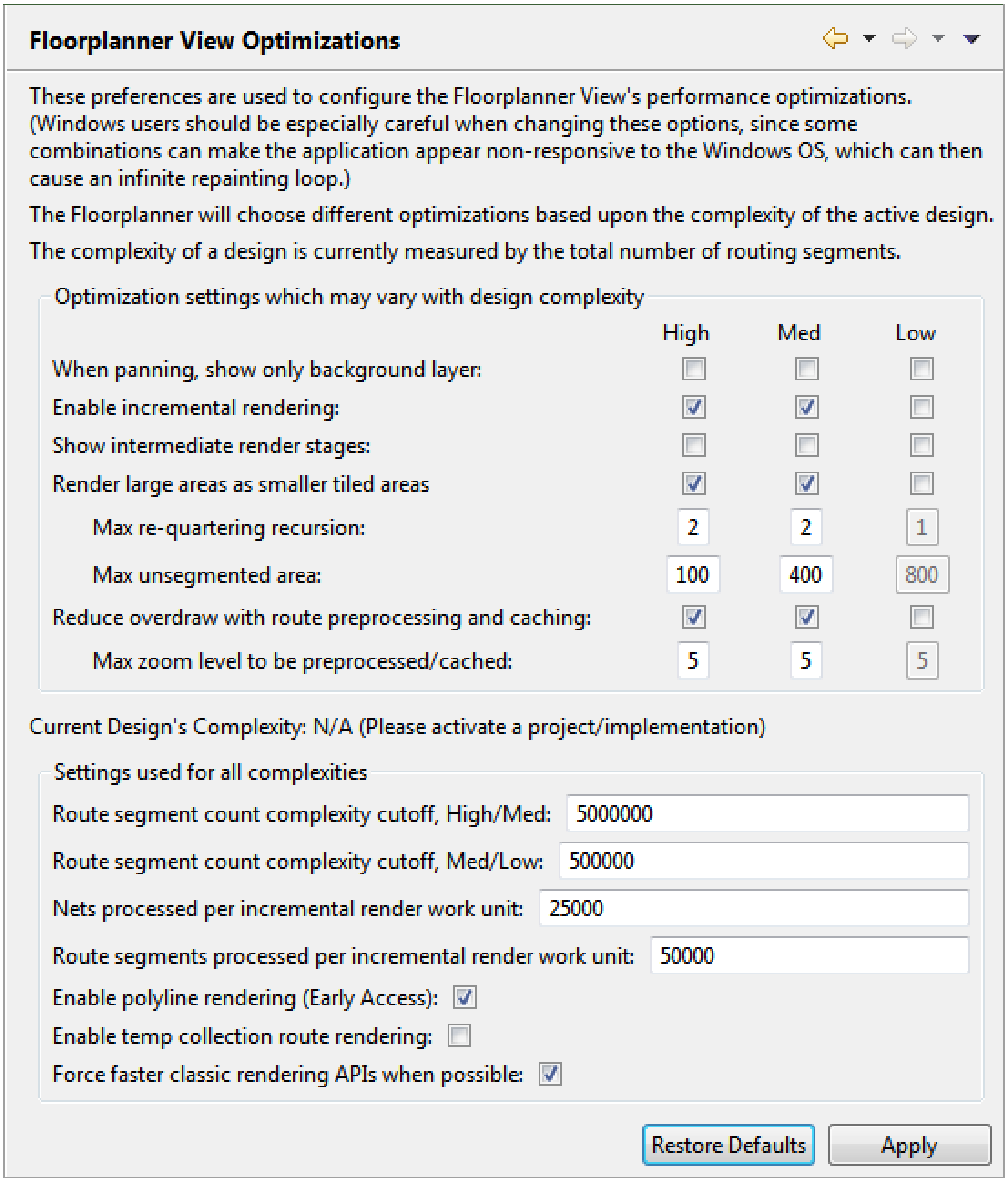


Figure 92: *Floorplanner View Optimizations Preference Page Example*

Designs on modern FPGAs are continuing to increase in complexity. To maintain acceptable Floorplanner performance, highly complex designs require significant rendering optimizations. The set of preferences on this page allows advanced users, FAEs, and tech support to tweak the Floorplanner optimization settings in rare cases when it proves necessary.

ACE tracks three levels of design complexity (High, Med, and Low) and, by default, enables or disables individual optimization settings based upon the design complexity. Because drawing the routing has the most significant impact upon Floorplanner render performance, design complexity is measured in terms of the route segment count. The cutoffs between complexity levels are user configurable.

> **Note**
>
> By default, all optimization features are disabled for the simplest designs. As design complexity increases, more optimizations are enabled by default. Some optimizations have overhead, and actually hinder render performance on small designs — for this reason, all optimizations are typically disabled for the simplest (Low complexity) designs.

The current (active) design Floorplanner complexity is always reported in this preference page as **Current Design's Complexity:** (near the middle of the page). This detail helps to determine which column of optimization settings impacts the rendering of the current design in the Floorplanner. Be aware that the route segment count used to compute the design complexity only includes route segments of routed nets. The same design often reports a Low complexity before it is routed, and a High complexity after routing is complete.

Table 116: *Optimization Settings*

Option	Technical Description	Usability Notes
Optimization settings which may vary with design complexity		
When panning, show only background layer:	Reduces the amount of rendering performed while panning / scrolling; the detailed render only occurs after panning / scrolling is completed.	Enable this if panning / scrolling the Floorplanner feels too slow.
Enable incremental rendering:	The render work is broken up into small chunks within each render layer and performed asynchronously, instead of performing the entire render of all layers at once. Because the work chunks are performed asynchronously, the application has a chance to respond to subsequent mouse and keyboard input earlier, instead of waiting for the entire render to complete.	If enabled, each Floorplanner render is slightly (5% to 10%) slower overall, but the Floorplanner Perspective becomes significantly more responsive to mouse and keyboard actions. Obsolete renders might then be interrupted (allowing faster renders when quickly changing zoom levels with the mouse wheel, for example). In some cases (for example, if a great deal of Floorplanner panning/scrolling occurs), there might be noticeable rendering latency/lag.
Show intermediate render stages:	When renders are slow, it can be frustrating to stare at an empty grey/black window waiting for something to change. When this setting is enabled, the user can observe as render layers are built up into the final rendered image.	Provides more frequent visual feedback during rendering, (so it can feel faster, because something is visibly happening,) but renders are actually slightly slower overall.
Render large areas as smaller tiled areas:	The total render area, if greater than the **Max unsegmented area**, are broken into quadrants which are rendered individually. Rendered areas are checked for (re-)quartering up to **Max re-quartering recursion** times. Because the quadrant area is smaller, it completes rendering faster than the whole.	Large render areas are each broken into four smaller chunks for increased visual feedback. Enabling this increases total render times, but it might feel faster, because something is visibly happening.
• **Max re-quartering recursion:**	The maximum number of times a render area may be broken into smaller (and smaller) pieces. Only relevant when **Render large areas as smaller tiled areas** is enabled.	Relevant at the outermost zoom levels. Increasing this value might increase total render times, but can provide more frequent visual feedback.

Option	Technical Description	Usability Notes
• **Max unsegmented area:**	Areas larger than this are broken into smaller chunks up to **Max re-quartering recursion** times. Only relevant when **Render large areas as smaller tiled areas** is enabled.	Relevant at the outermost zoom levels. Decreasing this can increase total render times, but can provide more frequent visual feedback.
Reduce overdraw with route preprocessing and caching:	Significantly improves render speeds by reducing route line overdraw via culling. Routing data is pre-processed and cached at multiple zoom levels when the routing data is loaded/updated. Due to memory constraints, only the outermost zoom levels may be cached.	By pre-processing routes at multiple zoom levels when the routes are loaded, we reduce the number of route lines we draw over preexisting routes lines of the same color. This slightly increases the memory footprint and load times, but significantly reduces render times for large designs (when overdraw is most frequent).
• **Max zoom level to be preprocessed/cached:**	0=zoomed all the way out, 1=zoomed in one step, etc.	This number must be kept small to avoid running out of memory. (Floorplanner route render cache sizes can more than double at each increasing zoom level.)
Settings used for all complexities		
Route segment count complexity cutoff, High/Med:	Designs with a route segment count greater than this number use the optimization settings for High complexity designs (the first column of checkboxes /fields).	–
Route segment count complexity cutoff, Med/Low:	Designs with a route segment count less than (or equal to) this number use the optimization settings for Low complexity designs (the third column of checkboxes/fields).	–
Nets processed per incremental render work unit:	The number of nets to process per discrete work unit when rendering the routing layers. Used when the current zoom is NOT in the route overdraw reduction cache. Only relevant when **Enable incremental rendering** is on.	Increasing this number might very slightly improve rendering performance, but decrease the frequency of visual feedback.
Route segments processed per incremental render work unit:	The number of route segments to process per discrete work unit when rendering cached route segments. Used when the current zoom is in the route overdraw reduction cache. Only relevant when both **Enable incremental rendering** and **Reduce overdraw with route preprocessing and caching** are on.	Increasing this number might very slightly improve rendering performance, but decrease the frequency of visual feedback.
Enable polyline rendering (Early Access)(1)	–	Polyline rendering of the routes is a known major Floorplanner performance advantage in Windows, but only minor advantages were seen in tested Linux configurations. The advantages are most noticeable in the largest designs.
Enable temp collection route rendering:(2)	–	Might slow rendering when using route overdraw reduction cache, but other route rendering might speed up slightly.
Force faster classic rendering APIs when possible:	–	The classic rendering APIs are significantly faster in all tested cases, but might produce slightly cruder visual output due to a lack of anti-aliasing. (The modern/advanced render APIs are still automatically used when absolutely required.)

Option	Technical Description	Usability Notes

Table Notes

1. **Caution: Enable polyline rendering** is new, Early Access functionality. While Achronix found no negative stability or performance impacts when testing this optimization, customer hardware and software have a wider variety of configurations than in our test labs. If any new performance or stability issues are observed in the GUI, please contact Achronix Technical Support. Achronix notes the specifics of your configuration (to reproduce and correct the problem), and might then suggest disabling this new polyline rendering functionality to determine if performance or stability improves.
2. **Caution: Enable temp collection route rendering** requires significantly more ACE GUI memory when enabled!

Technical Note for Windows Users

The Windows operating system requires that applications check-in every five seconds, or the application is deemed non-responsive. Non-responsive applications are given a figurative kick-in-the-pants by Windows, and asked to repaint the screen. When the screen paint itself is taking more than five seconds, as might happen with poor Floorplanner Optimization settings, an application can be forced into an effective infinite-loop of paint requests from the operating system.

If a Windows user ever notices ACE being called non-responsive by Windows (check the application title bar), ACE has most likely entered this looped painting state. To escape, change back to the Project perspective (or any other perspective without the Floorplanner view visible), then on this Floorplanner View Optimizations Preference Page, ensure that incremental rendering and tiled rendering are both enabled for the current design's complexity level. If both are already enabled, please call Achronix Technical Support for guidance on further Floorplanner optimization tweaks.

Warning!

Disabling optimizations that are enabled by default is not recommended.

I/O Designer Preference Page

The I/O Designer Preference Page contains a number of preferences for the I/O Designer views.

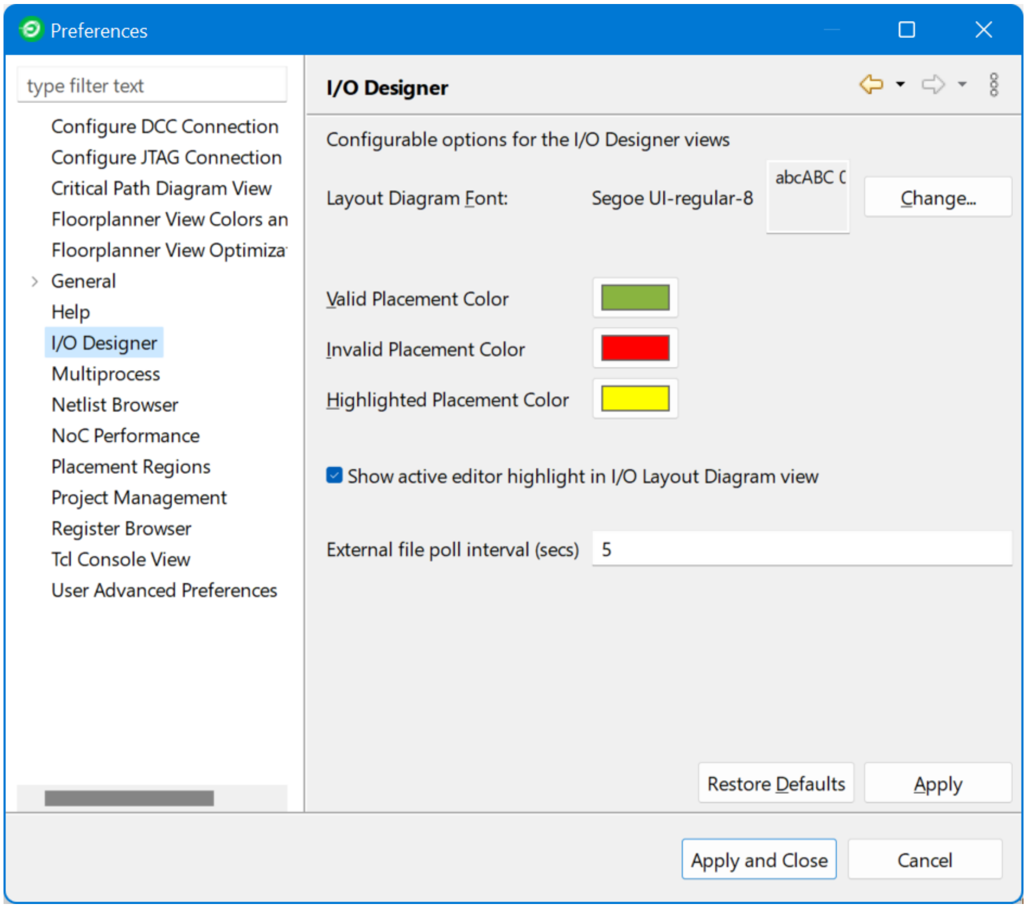


Figure 93: ***I/O Designer Preference Page Example***

Table 117: ***I/O Designer Preferences Options***

Option	Description
Layout Diagram Font	The font used on shapes in the I/O Layout diagram.
Valid Placement Color	The color used for sites with valid placements.
Invalid Placement Color	The color used for sites with invalid placements.
Highlighted Placement Color	The color used to highlight the currently selected site.
Show active editor highlight...	Toggles whether the currently selected site is drawn with a highlight.
External file poll interval (secs)	Specifies how often IP files are checked for external modification.

IP Diagram Preference Page

The IP Diagram Preference Page contains a number of preferences for the IP Diagram View (see page 80) relating to colors and fonts.

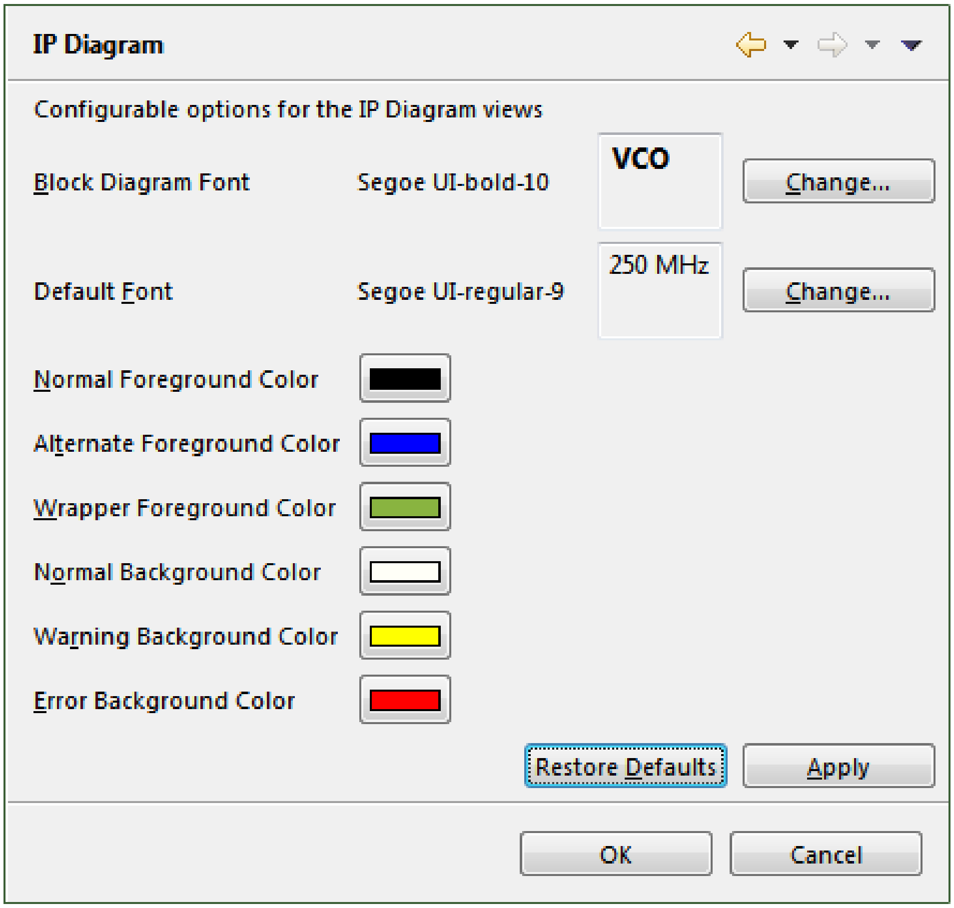


Figure 94: ***IP Diagram Preference Page Example***

Table 118: ***IP Diagram Preferences Options***

Option	Description
Block Diagram Font	The font used to title logic blocks in the diagram.
Default Font	The font used for all diagram text except the logic block titles.
Normal Foreground Color	The color used for logic blocks, signals, and text.
Alternate Foreground Color	The color used to highlight portions of the diagram.
Wrapper Foreground Color	The color used to represent the IP RTL wrapper itself. Everything enclosed by this color is within the wrapper.

Option	Description
Normal Background Color	The default background color for the entire diagram.
Warning Background Color	Text representing IP Options with warnings have their backgrounds painted this color.
Error Background Color	Text representing IP Options with errors have their backgrounds painted this color.

Multiprocess: Configure Custom Job Submission Tool Preference Page

The Multiprocess: Configure Custom Job Submission Tool Preference Page allows configuring the ACE Multiprocess View (see page 83) to submit Multiprocess jobs to a third-party cloud/grid/job submission system using a command-line tool. As a useful example, by default, ACE is configured to use an Oracle Grid Engine derivative via the `qsub` command (see https://en.wikipedia.org/wiki/Oracle_Grid_Engine). Be aware that the Oracle Grid Engine `qsub` arguments are not 100% compatible with the `qsub` standard (documented at http://pubs.opengroup.org/onlinepubs/9699919799/utilities/qsub.html).

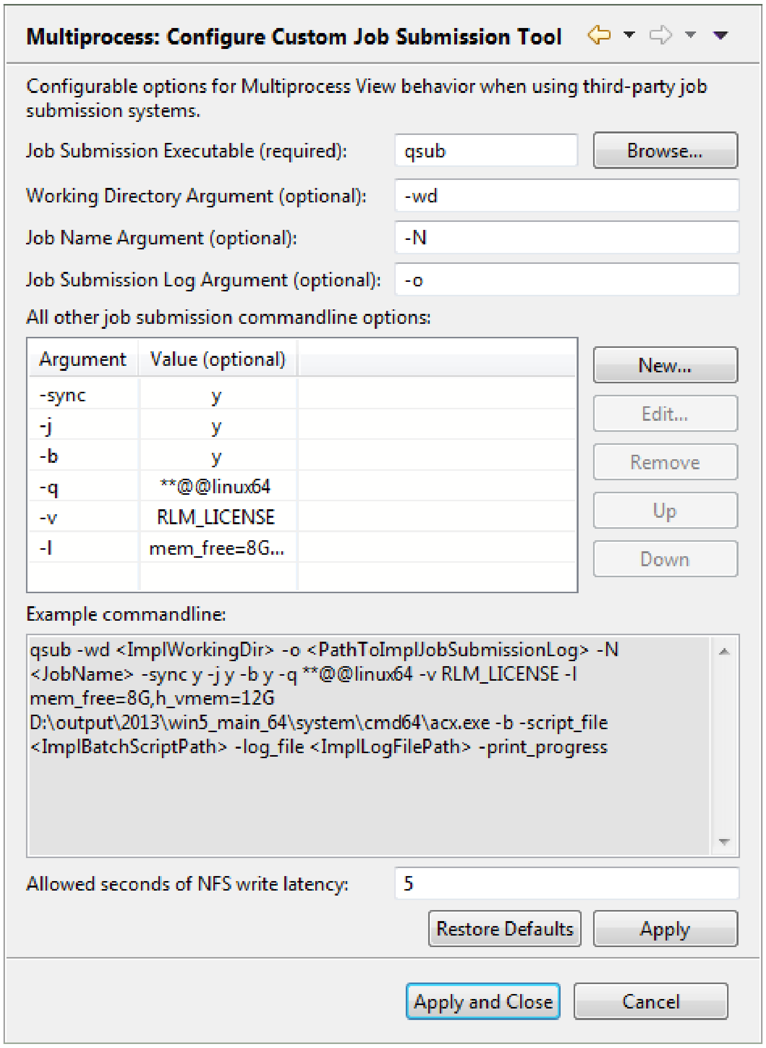


Figure 95: ***Multiprocess: Configure Custom Job Submission Tool Preference Page Example***

When the third-party job submission system support is enabled, ACE calls the specified executable, using the specified command-line arguments, to submit ACE Multiprocess jobs.

See Configuring ACE to use an External Job Submission System (see page 285) for more information.

Warning!

Debugging job submission configurations:

- If the job submission system is properly configured on the host machine (meaning it is possible to successfully submit non-ACE jobs from the command-line) and ACE is still unable to successfully submit jobs, please contact Achronix technical support.
- **DO NOT** copy the text of the attempted command and manually attempt to submit from the command-line.

Netlist Browser Preference Page

The Netlist Browser preference page contains a number of preferences for the Netlist Browser view.

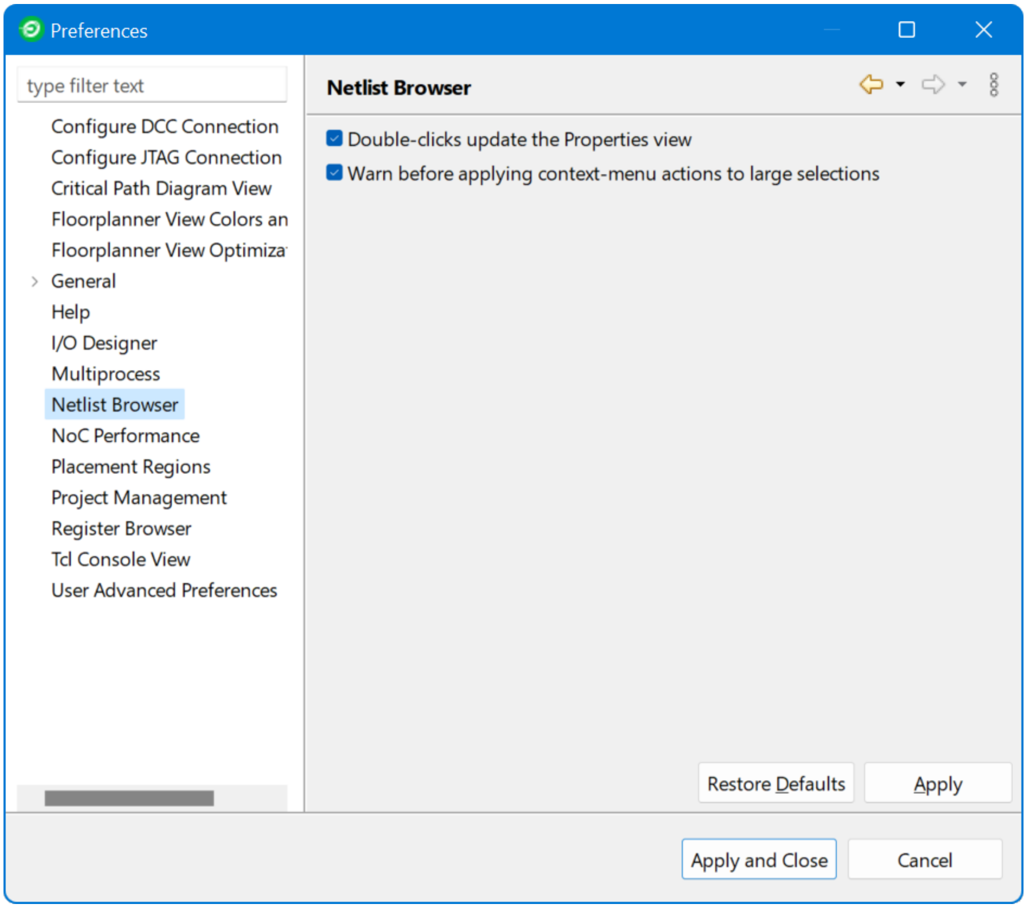


Figure 96: *Netlist Browser Preference Page Example*

Table 119: *Netlist Browser Preferences Options*

Option	Description
Double-clicks update the Properties view	Toggles whether double-clicking an item in the Netlist browser view shows that item in the Properties view.
Warn before applying context-menu actions to large selections	Applying actions to very large selections can take a very long time. Toggling this option causes a warning message to appear before an action is run against a large selection.

NoC Performance View Preference Page

The NoC Performance view preference page offers preferences related to the 2D NoC Performance view.

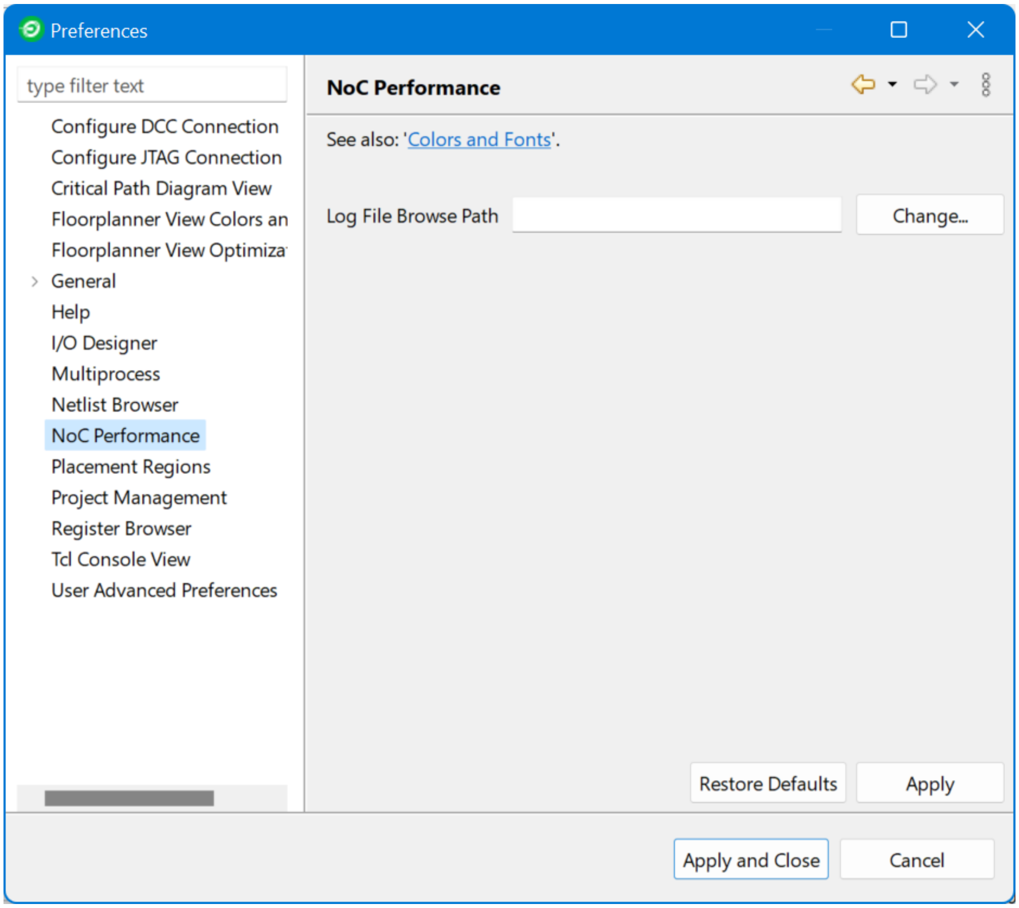


Figure 97: ***NoC Performance View Preference Page***

Table 120: ***NoC Performance View Preferences***

Preference	Description
Log File Browse Path	By default, browsing for simulation log files starts in whatever directory was last browsed. Setting the log file directory path causes browsing to always start at the specified path.

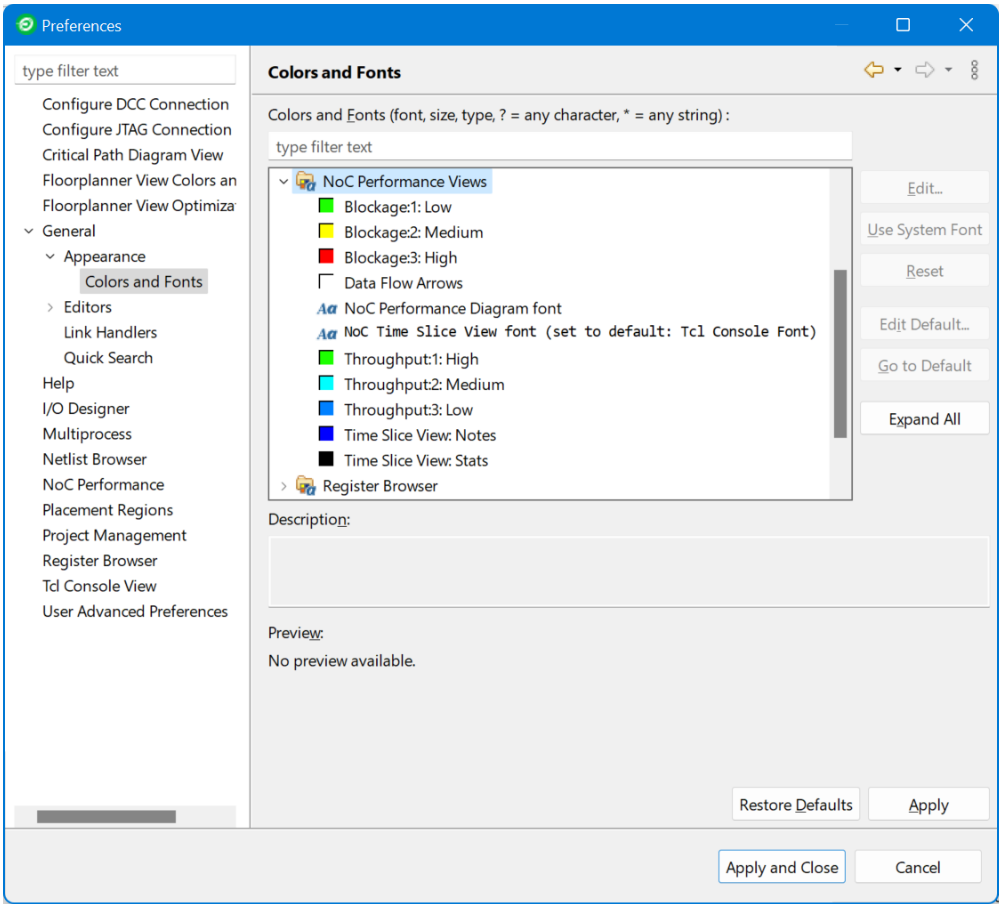


Figure 98: ***Colors and Fonts | NoC Performance View Section***

Table 121: ***Color Preferences***

Color Preference	Description
Blockage:1:Low	Selects the color to render shapes with "low" blockage.
Blockage:2:Medium	Selects the color to render shapes with "medium" blockage.
Blockage:3:High	Selects the color to render shapes with "high" blockage.
Data Flow Arrows	Selects the color to render "data flow direction" arrows.
NoC Performance Diagram font	Selects the font used to label diagram shapes.
Noc Time Slice View font	Selects the font used to render text in the NoC Time Slice view
Throughput:1:High	Selects the color to render shapes with "high" throughput.
Throughput:2:Medium	Selects the color to render shapes with "medium" throughput.

Color Preference	Description
Throughput:3:Low	Selects the color to render shapes with "low" throughput.
Time Slice View: Notes	Selects the color to render "notes" in the Time Slice view.
Time Slice View: Stats	Selects the color to render "stats" in the Time Slice view.

Other Colors and Fonts Preference Page

The Other Colors and Fonts Preference Page allows setting many of the fonts, colors and components used by ACE. The page is accessed by selecting **General** → **Appearance** → **Other Colors and Fonts**.

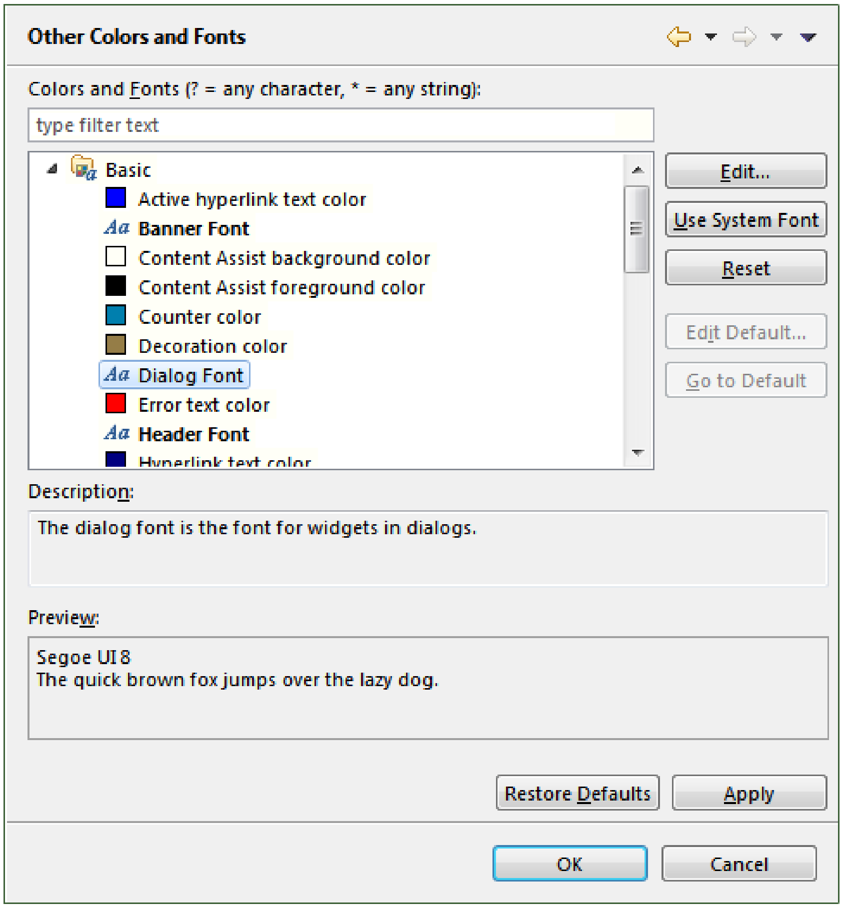


Figure 99: *Other Colors and Fonts Preference Page Example*

A tree is used to navigate among and show a short preview of the various colors and fonts. The current face (but not size) of any font is previewed in its label. Colors are previewed in the icon associated with its label. Additionally, some categories (Workbench in particular) provide a more detailed preview of their contributions (shown below the description area, if available).

- Font settings can be changed either by selecting the font from the list and clicking **Use System Font** to use the operating system font, or by clicking **Edit** to open up a font selection dialog. **Reset** can be used to return to the default value.
- Color settings can be changed by clicking **Edit** to the right of the tree area when a color is selected. **Reset** can be used to return to the default value.
- The **Colors and Fonts** text field can be used to filter the contents. Simply type in an entry and any matching results remain in the tree view.
- The **Description:** and **Preview:** sections provide details when the Workbench colors and font settings are selected.

Package View Preference Page

The Package View Preference Page provides color configuration for several graphics layers in the Package view (see page 112).

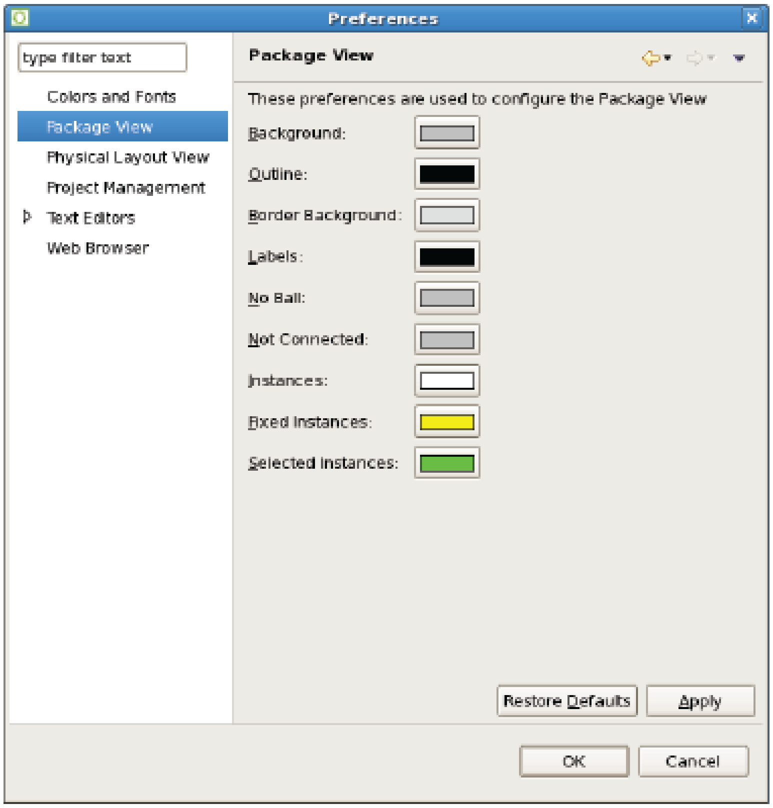


Figure 100: ***Package View Preference Page Example***

Placement Regions Preference Page

The Placement Regions Preference Page determines how Placement Regions (see page 354) are handled in the Placement Regions view (see page 120) and the Floorplanner view (see page 53) (when the Floorplanner Placement Region Tool is active).

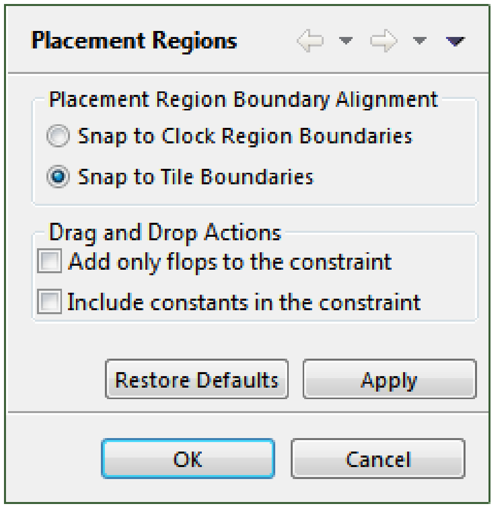


Figure 101: ***Placement Regions Preference Page Example***

Table 122: ***Placement Region Preferences***

Option	Description
Placement Region Boundary Alignment	
Snap to Clock Region Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries are forced to align with Clock Region Boundaries. Use this for a simple, coarse-grained approach to Placement Regions. This setting is recommended for most use cases.
Snap to Tile Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries are forced to align with Tile Boundaries, for a very fine-grained region. This setting is only recommended for advanced users.
Drag and Drop Actions	
Add only flops to the constraint	When using drag-and-drop to assign Placement Region Constraints, this setting ensures only flops are assigned to the region constraint. All other dropped items are excluded from the constraint.
Include constants in the constraint (1)	When using drag-and-drop to assign Placement Region Constraints, this setting includes constants in the constraint.

Table Notes

1. The **Include constants in the constraint** setting is ignored if **Add only flops to the constraint** is enabled.

Project Management Preference Page

The Project Management Preference Page sets the behavior of Editors (see page 26) and Reports (see page 227).

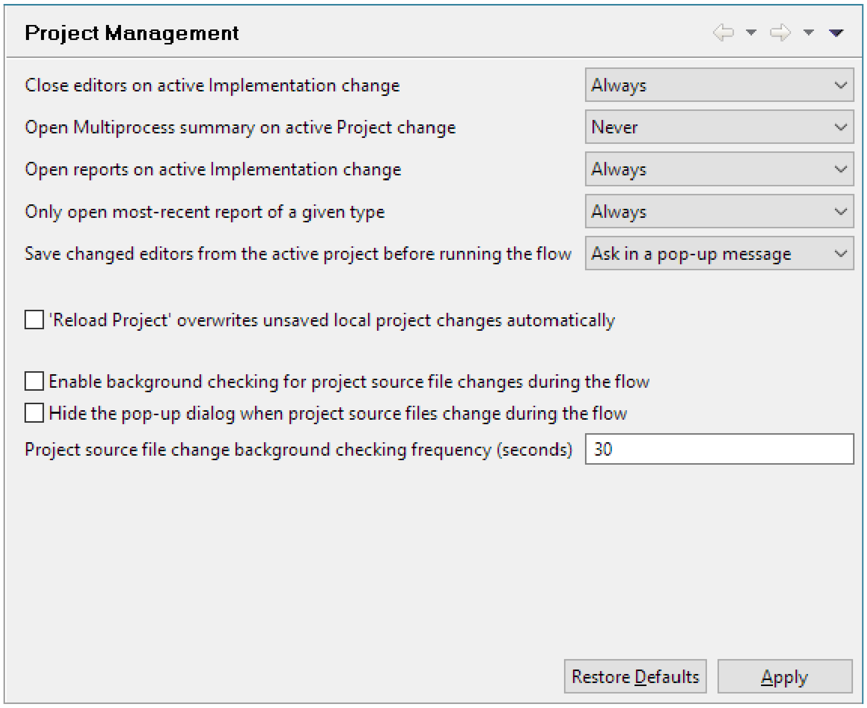


Figure 102: ***Project Management Preference Page Example***

Tcl Console View Preference Page

The Tcl Console View Preference Page contains settings that alter the behavior and/or presentation of information in the Tcl Console View (see page 144).

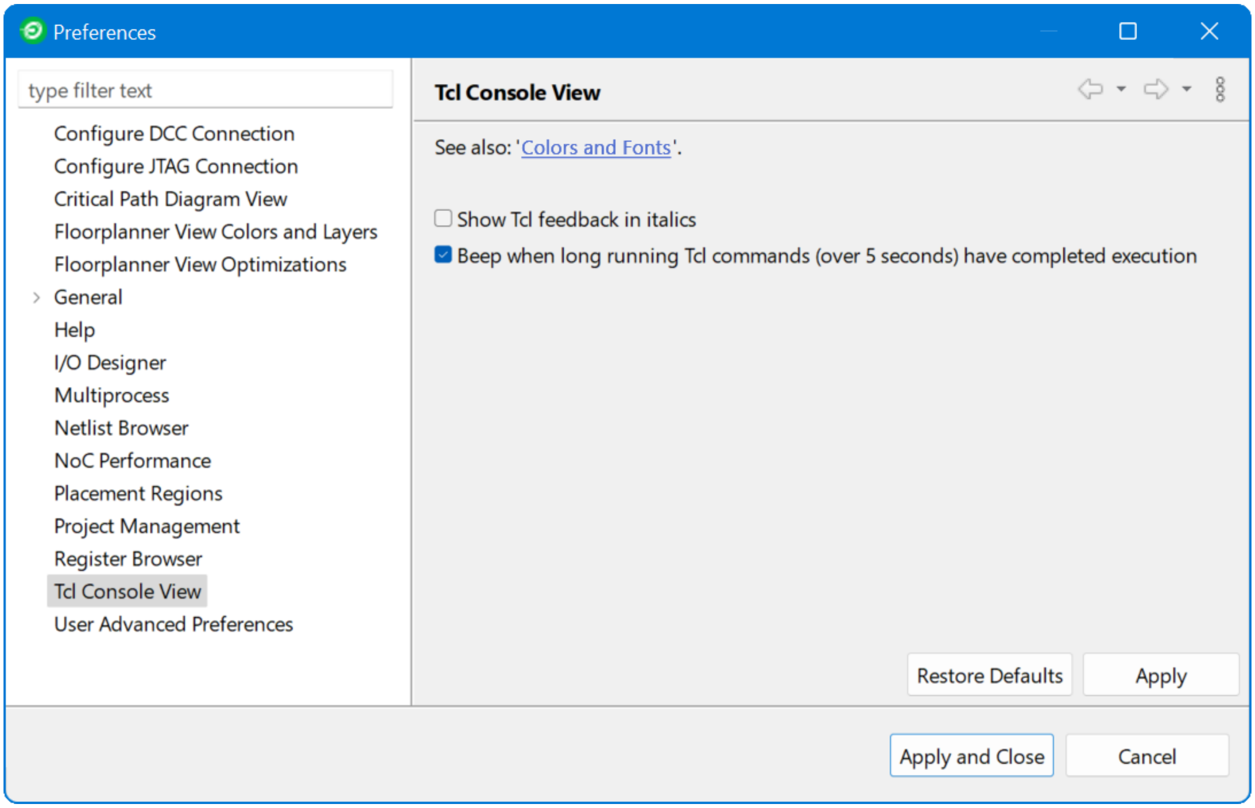


Figure 103: ***Tcl Console View Preference Page Example***

Table 123: ***Tcl Console View Preferences***

Option	Description
See also:	Link to Colors and Fonts Preferences allowing choice of the font used in the Tcl Console.
Show Tcl feedback in italics	Enabling this option causes the Tcl console feedback to be shown in italics.
Beep when long running Tcl commands (over 5 seconds) have completed execution	Enabling this option provides audible feedback (the default system beep/bell sound) upon completion of long-running commands.

Text Editors Preference Page

The Text Editors Preference Page sets the behavior and appearance of the text editor.

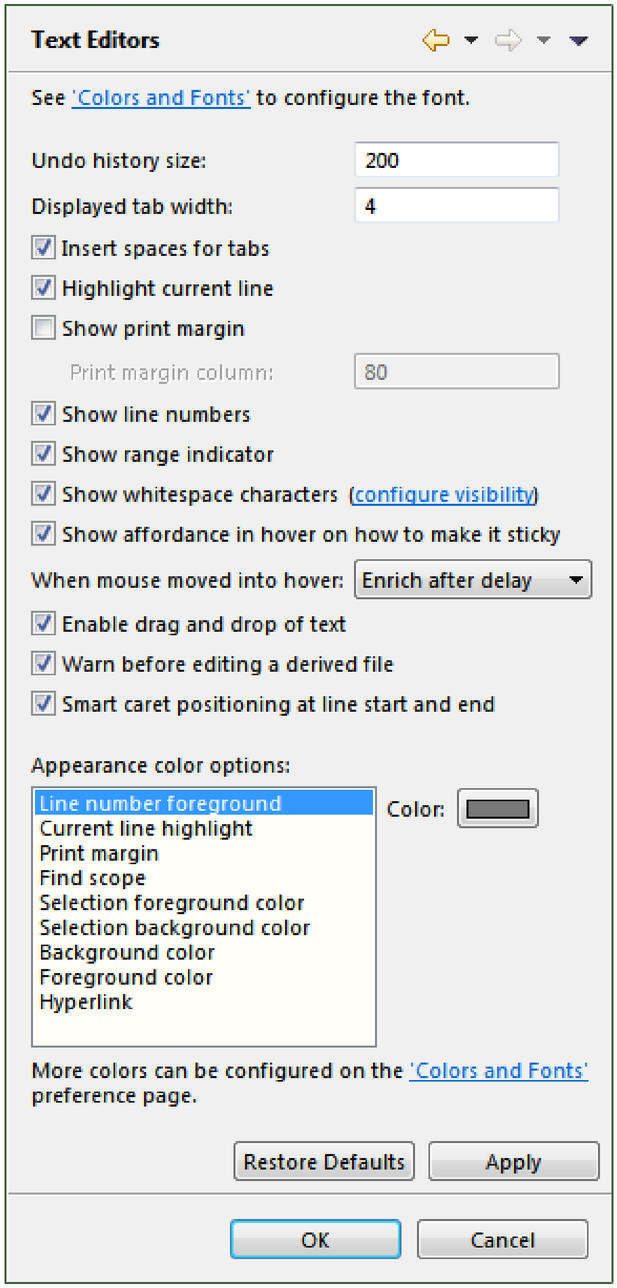


Figure 104: ***Text Editors Preference Page Example***

Table 124: *Text Editor Options*

Option	Default	Description
Undo history size	200	Sets the number of undo events in the history queue.
Display tab width	4	Sets the editor tab width in spaces.
Insert spaces for tabs	Deselected	Enables insertion of spaces in place of tab characters.
Highlighting current line	Selected	Enables/disables the highlighting of the current line. The highlight color is set in **Appearance color options**.
Show print margin	Deselected	Selects whether the print margin is visible.
Print margin column	80	Sets the print margin column position.
Show line numbers	Selected	Enables/disables the display of line numbers in the Editor view.
Show range indicator	Selected	Enables the display of range indicators in the text editor.
Show whitespace characters	Deselected	Enables the display of whitespace characters (·) in text editors.
Show affordance in hover on how to make sticky	Selected	Enable the affordance (visual clue) in the hover text and make it sticky.
When mouse moved into hover	Enrich after delay	Sets the hover display mode.
Enable drag and drop of text	Selected	Enables/disables the ability to drag and drop selected text.
Warn before editing a derived file	Selected	Enables warning if a derived file is going to be edited.
Smart caret position at line start and end	Selected	Controls whether the editor automatically positions the caret and the start or end of a line.
Appearance color options	Various	Sets custom colors for various aspects of the text editor.

Quick Diff Preference Page

The Quick Diff Preference Page, enables the Quick Diff option and configures its appearance. The page is accessed via **Text Editors → Quick Diff**.

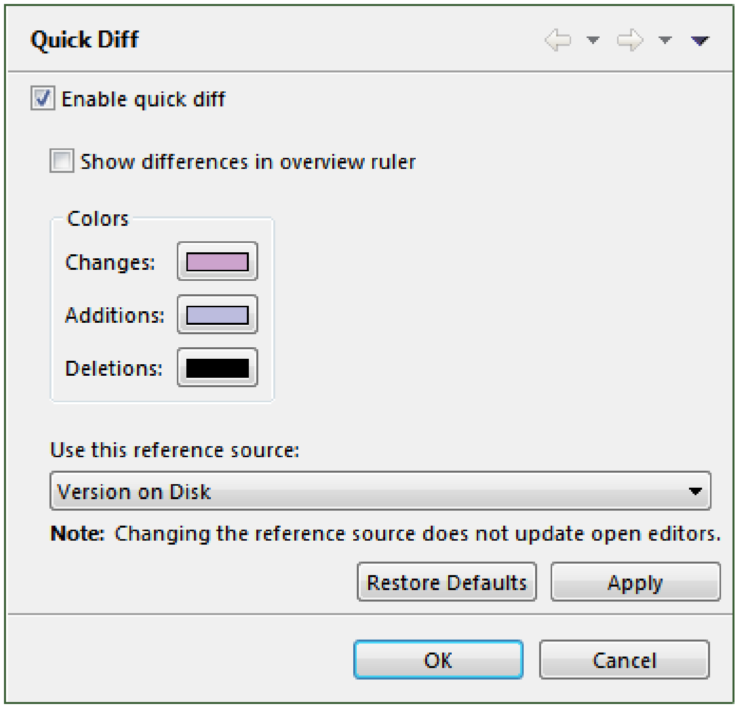


Figure 105: ***Quick Diff Preference Page Example***

Table 125: ***Quick Diff Preference Page***

Option	Default	Description
Enable quick diff	Selected	Enables/disables the quick diff option.
Show differences in overview ruler	Deselected	Shows differences in the overview ruler.
Colors		
Changes	–	Sets the color used to indicate changes. (1)
Additions	–	Sets the color used to indicate additions. (1)
Deletions	–	Sets the color used to indicate deletions. (1)
Use this reference source	Version on disk	This option sets which reference to use as the base for generating quick diff comparisons. Options are: **Version on Disk**: Current file is compared against the last saved version on disk.

Option	Default	Description
Table Notes 1. The button to the right of the option allows changes to the display color (refer to "Changing Color Coding").		

Projects

A project represents the collection of source netlist and constraints files, flow options, IP configuration files, and output files for a particular design.

Implementations

A Project (see page 215) may have multiple implementations. Each implementation contains the set of flow options (also called implementation options) configuring the run of that project through the Flow (see page 221), and the flow outputs for this particular configuration. With this capability, the same design (netlist) can be implemented (run through the flow) with different sets of timing constraints, placement and routing optimizations, or even different target devices, just by creating multiple implementations for the same project.

Each implementation is associated with an implementation directory located under the project directory (where the Project File (see page 216) is located). Implementation directories are named with the implementation name and contain flow output files. Output Files (see page 218) are divided into two sub-directories under the implementation directory: `output` and `reports`. The `output` directory contains files that are intended to be consumed by other tools later in the flow, such as netlists for simulation or the FPGA bitstream for programming. The `reports` directory contains files intended to be viewed and analyzed while running the ACE flow, such as timing reports and flow statistics.

Implementation definitions are *not* individually saved to their own files. Instead they are stored as part of the Project File (see page 216). In the GUI, project implementations can be browsed in the Projects View (see page 125). Selecting an implementation activates (see page 221) it and displays its implementation options in the Options View (see page 103).

When an implementation has been run through the flow, the state of the database (netlist, constraints, placement, and routing data) may be saved to an `.acxdb` file (see Saving Implementations (see page 276)). Implementations may later be restored from previously saved `.acxdb` files (see Restoring Implementations (see page 277)).

Implementation Options

There are a wide variety of configurable implementation options which alter how ACE processes that implementation of the design as it moves through the flow (see page 221). The the most-commonly used option settings are displayed in the Options View (see page 103) for the current Active Project and Implementation (see page 221). Within the Options view, implementation options are grouped by flow steps (see page 221) to indicate which flow step the option affects. Changing the value of an option causes the current results of that flow step (if any) to become invalid or cleared, and that flow step (and all later flow steps) must be rerun, making use of the newly-changed option.

An Implementation Options Report (see page 240) of all available implementation options may be generated via the Tcl command report_impl_options (see page 578). This command may also be used to compare the current options configuration of an implementation with the default values for all options.

The values of implementation options may be set with the Tcl command set_impl_option (see page 602), or reset back to the default values with reset_impl_option (see page 583).

Option Sets

Because some implementation options have a large impact upon runtime, and because the QoR benefits of these implementation options may vary significantly by design (often a QoR gain, but sometimes a slight QoR loss), many of the performance-related implementation options are disabled by default for newly created projects and implementations.

Achronix QoR experts have compiled subsets of implementation options known to optimize a wide variety of design types based upon design details. These "option sets" are made available (with description) to users in the Multiprocess View (see page 83), and through that view, may be used to generate new implementations with the indicated implementation options enabled.

Each Option Set shown in the Multiprocess View consists of override values for a small subset of all the implementation options. These overriding values are applied to newly generated implementations over the existing implementation option values inherited from a user-selected template implementation.

It is worth repeating that the Option Sets do not contain a complete assignment of all the implementation options. Each Option Set only contains a small subset of option values, which override the implementation options inherited from the template implementation. The overriding implementation options in each set are **subsets** of the entire set of QoR oriented implementation options. They only change *some* of the implementation options, and all the rest of the values are inherited from the template implementation. The Option Sets only enable performance-related implementation options, and (currently) never disable any already-enabled implementation options. So each generated implementation starts with the exact same implementation options as the template implementation, and then just the few implementation options named in the Option Set description are overwritten with the described values.

Achronix broke up the Option Sets into small granular chunks because of QoR/runtime trade-offs. Some of the options have a large runtime cost, and on some designs, there is a minimal performance gain. Based upon the observed runtimes reported in the Multiprocess Summary Report (see page 238), hours of runtime may be saved while only losing 0.01% frequency by using one Option Set over another Option Set as a design is iterated.

> **Note**
>
> Currently, the Option Set overrides only *enable* optimization-oriented implementation options, not disable them. Thus, if the implementation options in the template implementation are already the same values as those in the Option Set, the results from the two implementations (the implementation generated from the Option Set, and the template implementation) are identical.

It is expected that among all the Option Sets, at least one can be found that provides the necessary QoR gain for an acceptable runtime impact, allowing the fastest possible design iteration. See the Multiprocess View (see page 83) and Attempting Likely Optimizations Using Option Sets (see page 352) for more information.

Project File

Projects are persisted in `.acxprj` project files created automatically by the tool whenever a project is saved. A project file is actually just a Tcl script supporting only a defined subset of Tcl commands. Project files can be edited manually and then loaded into the tool to use as a script or for running regressions.

> **Note**
>
> When ACE loads a project file, it locks that file to prohibit other ACE sessions from loading the same file. This prevents project data corruption, which could occur if two sessions attempt to work with the same project at the same time.

In the GUI, loaded project file contents are displayed in a tree structure in the Projects View (see page 125). Project file contents may also be viewed in a Text Editor (see page 28) in the GUI by double-clicking the project name in the Projects view (example file contents follow):

```
Example Project file contents

# proj2
# AUTOMATICALLY GENERATED FILE
# MAY BE OVERWRITTEN AT ANY TIME DURING USE OF TOOL
# Netlist Files
add_project_netlist -project proj2 "C:/test_projects/proj2/top.vma"
# Constraint Files
add_project_constraints -project proj2 "C:/test_projects/proj2/clock_mode2.sdc"
add_project_constraints -project proj2 "C:/test_projects/proj2/clock_mode1.sdc"
# Implementations
# impl_1
create_impl -project proj2 impl_1
set_impl_option -project proj2 -impl impl_1 partname ACDevice1
set_impl_option -project proj2 -impl impl_1 speed_grade "standard"
set_impl_option -project proj2 -impl impl_1 core_voltage "1.00"
enable_project_constraints -project proj2 -impl impl_1 "C:/test_projects/proj2/clock_mode2.sdc"
disable_project_constraints -project proj2 -impl impl_1 "C:/test_projects/proj2/clock_mode1.sdc"
# impl_2
create_impl -project proj2 impl_2
set_impl_option -project proj2 -impl impl_2 partname ACDevice1
set_impl_option -project proj2 -impl impl_2 speed_grade "standard"
set_impl_option -project proj2 -impl impl_2 core_voltage "0.95"
enable_project_constraints -project proj2 -impl impl_2 "C:/test_projects/proj2/clock_mode2.sdc"
enable_project_constraints -project proj2 -impl impl_2 "C:/test_projects/proj2/clock_mode1.sdc"
# impl_3
create_impl -project proj2 impl_3
set_impl_option -project proj2 -impl impl_3 partname ACDevice1
set_impl_option -project proj2 -impl impl_3 speed_grade "standard"
set_impl_option -project proj2 -impl impl_3 core_voltage "0.95"
disable_project_constraints -project proj2 -impl impl_3 "C:/test_projects/proj2/clock_mode2.sdc"
disable_project_constraints -project proj2 -impl impl_3 "C:/test_projects/proj2/clock_mode1.sdc"
# End of file
```

Source Files

A project contains source files used as inputs to the ACE flow. There are two types of source files:

1. Synthesized netlist files
2. SDC/PDC/PRT constraints files

In the GUI, source files may be browsed in the Projects View (see page 125) and viewed in the built-in Text Editor (see page 28) by double clicking the file name in the Projects View.

The synthesized netlist files must be the gate-level Verilog output of the Synthesis tool and must use a `.v` or `.vma` file extension.

The constraints file types are defined in the following table.

Table 126: *Project Source File Types*

File Type	Description
`*.sdc`, `*.scf`	Timing constraints files in SDC format. These files are read by the timer in ACE. All timing constraints must be placed in these files.

File Type	Description
*.pdc	Placement constraints files for ACE. ACE can support many functions in this type of file, including placement and insertion of boundary pins. No timing constraints can be placed in these files.
*.prt	Partition definition file for incremental compile in ACE. There should be only one *.prt file per project in ACE. This file is generated by Synplify and controls which partitions will be re-compiled in the next ACE run.

Note

All *.sdc, *.scf, and *.pdc constraints files are read in and executed in the order specified in the ACE project file. Constraints files that are added to the project first are executed first, and likewise, constraints files which are added to the project last are executed last. If there is any order dependency between commands in your constraints, please make sure to add the constraints files in the correct order for execution in ACE.

IP Configurations

ACE provides GUI support to ease configuration of the most complicated embedded IP in Achronix FPGAs. The data files used by the IP Configuration Editors (see page 26) (files with the .acxip extension) may optionally be associated with a project. When associated with a project, these IP Configuration files may then be browsed in the Projects view (see page 125) under the project IP folder, and the associated editor may be started by double clicking the file name in the Projects View.

For more details, see Creating an IP Configuration (see page 306), or one of the many IP Configuration Editors.

Port Mapping Files

The I/O Designer I/O Pin Assignment View (see page 69) and I/O Core Pin Assignment View (see page 70) provide the option to remap port names from the values used in the IP editors to the name desired for use in the user RTL (top-level port list). This action creates an "alias" to be used in all files created when the **Generate IORing Design Files** button in the IO Designer is clicked (or similarly, calling the generate_ioring_design_files Tcl command).

These remapped names are placed in the board.acxpm (remapped pin ports) and core.acxpm (remapped core pin ports) files, stored in the project directory (the directory containing the *.acxprj Project file (see page 216)). These files are later read when generating the design files and are not needed within the design/netlist, nor are they explicitly named in the *.acxprj Project file (see page 216).

These are simple text files, with the contents automatically managed by the I/O Pin Assignment View (see page 69) and I/O Core Pin Assignment View (see page 70). Each line in the port mapping file contains a key/value pair, of the form:

```
original_port_name=remapped_name_alias
```

See Creating an IP Configuration (see page 306) for more details on handling IP configurations and the I/O Designer.

Output Files

Output files are generated for project implementations (see page 215) by running certain steps in the Flow (see page 221). By default, output files are automatically written to the project implementation directory under the output or reports directory as appropriate. Alternate output locations can be specified when running individual flow steps (see page 221) with command line options.

In the GUI, output files can be browsed in the Projects view (see page 125) under their implementation and viewed in the editor area by double clicking the file name in the Projects view.

Log Files

A number of log files are automatically generated while ACE is running. They include the following:

- ACE Session Log
- Implementation Log
- Multiprocess Log
- SnapShot Log
- ACE GUI Log

The contents of these logs are typically the series of Tcl commands and resulting return values occurring during the execution of ACE. Achronix support may sometimes request one or more of these log files to assist in requested support efforts.

ACE Session Log

Every time ACE is started, a new ACE session log is created in the directory `<user_home_dir>/.achronix/`. This file is named `ace_<date_timestamp>.log`, where `<date_timestamp>` is <year>_<month>_<day>_<hour>_<minute>_<second>, in 24-hour format. For example, if ACE was started in Linux with a username of `example_user`, on January 11th of 2023 at 2:34:56 PM, the complete log file name would be:

```
/home/example_user/.achronix/ace_2023_01_11_14_34_56.log
```

ACE session log messages are also sent to the Tcl Console View (see page 144). For more information, see Viewing the ACE Log File (see page 304).

Implementation Log

In addition to the session log, each project (see page 215) implementation (see page 215) has a log maintained for the complete life of the implementation. All changes to the implementation, including running the Flow (see page 221) for that implementation, are appended to the implementation log. This log is stored in the directory `<project_dir>/<impl_name>/log/impl.log`.

Multiprocess Log

Unlike normal flow executions, implementation runs initiated from the Multiprocess View (see page 83) do not have their log information appended to the ACE Session Log. The reason is because multiple processes would be appending info to the log file simultaneously, leaving log entries interleaved in an unreadable mess. Instead, each implementation executed in the background creates a new log file named `multiprocessImpl.log` in the log directory for that implementation, overwriting any prior multiprocess log created for that implementation. Each implementation executed via the Multiprocess View does still append information to its lifetime implementation log file.

When running Multiprocess with an external job submission system (i.e., the example GridEngine default configuration), two additional log files may be created:

- The `jobExecution.log` contains a raw (unfiltered) copy of the implementation log for that multiprocess session, plus any additional info that the user custom job submission system might choose to inject. This file only exists if the (optional) job submission logging functionality is configured appropriately.
- The `jobScheduler.log` is only used during Multiprocess Batch Mode (see page 292) (when in GUI mode instead of batch mode, similar info is captured in the individual implementation feedback tabs within the Multiprocess View). This file captures the standard output and error streams from the job submission system itself. This log file is particularly useful when initially configuring ACE to work with the user job submission system as it captures submission configuration errors (such as typos in job queue names).

Snapshot Log

The Snapshot log file is discussed in Collecting Samples of the User Design (see page 346).

ACE GUI Log

On rare occasions, Achronix Support may request the ACE GUI Log. This log file may be found by selecting: **Help** → **About Ace** → **Installation Details** → **Configuration** → **View Error Log**, which opens the GUI log in a non-ACE text file editor or HTML browser. The editor/browser typically reports the full path of the opened file.

Active Project and Implementation

The active project is the project containing the active implementation in the current tool session. The active implementation is the project implementation on which flow and project management commands are operating. Only one implementation can be active at a time. The active state applies across all projects and only in the context of the current tool session. This state does not persist across sessions and is not saved in a project file.

In the ACE GUI, the active project name, active implementation name, and target device name are all shown on the ACE titlebar in the format "Project -> Implementation (Device)". Additionally, within the Projects View (see page 125) tree, the active project and its active implementation are both shown in a bold font.

Flow

The flow is the set of steps that must be run to complete a design in ACE. These steps are listed, in order, within the Flow View (see page 61). The current Flow Mode implementation option (selected in the Options View (see page 103)) affects which Flow Steps may be executed.

A flow can only be run on a single Project (see page 215), and within that project a single Implementation (see page 215) at a time (these are the Active Project and Implementation (see page 221)) during an interactive ACE session.

To run multiple implementations from the same project through the flow simultaneously, use the Multiprocess View (see page 83).

To run multiple separate projects through the flow simultaneously, multiple sessions of ACE must be run.

Additional details may be found in the section Running the Flow (see page 280).

Flow Steps

The Flow (see page 221) is composed of a series of flow steps, each representing a command operating on the design in the Active Project and Implementation (see page 221). Some flow steps are required (such as preparing the design), and some are optional (such as writing out a netlist for simulation). Flow steps are generally order-dependent, and running the steps out of order may result in errors.

The default order of flow steps is displayed in the ACE GUI Flow View (see page 61), with flow steps grouped into categories for organizational purposes.

The implementation option for Flow Mode (see page 226) (typically configured through the Options View (see page 103)) affects which flow steps can be executed.

Table 127: *Achronix Default Flow Steps and IDs*

Name	ID
Prepare	`prepare`

Name	ID
- Run Prepare	`run_prepare`
- Run Estimated Timing Analysis	`report_timing_prepared`
- Generate Pre-Placed Simulation Netlist	`write_netlist_prepared`
Place and Route	`place_and_route`
- Run Place	`run_place`
- Run Post-Placement Timing Analysis	`report_timing_placed`
- Run Route	`run_route`
- Run Post-Route Timing Analysis	`report_timing_routed`
Design Completion	`design_completion`
- Post-Process Design	`post_process`
- Run Final DRC Checks	`final_drc_checks`
- Run Sign-off Timing Analysis	`report_timing_final`
- Generate Final Reports	`write_reports_final`
- Generate Final Simulation Netlist	`write_netlist_final`
FPGA Programming	`fpga_program`
- Generate Bitstream	`write_bitstream`
- FPGA Download	`fpga_download`

Table Notes

- All flow step IDs can be executed at the ACE GUI Tcl console (see Tcl Console View (see page 144)) or as part of the user Tcl script that can be invoked when running ACE (see page 259) in batch mode. The following Tcl command allows executing the various flow steps IDs listed:
```
run [-step <string>] [-stop_at_step <string>] [-resume] [-ic <string>]
```
- Because advanced users are allowed to create their own flow steps (create_flow_step (see page 541)), this list may be a subset of the flow steps available to users. To see a complete list of current flow step IDs, use the Tcl command get_flow_steps (see page 559).

Prepare Steps

Run Prepare

The first flow step required for any design is **Run Prepare**. This step (in order):

1. Clears all previously loaded netlists and constraints.
2. Loads and compiles the device.
3. Loads all the design files for the active implementation into ACE.
4. Runs design checks.
5. Transmutes the design into an Achronix design.

The active project is saved to disk automatically when this step is successfully completed. In addition, this step automatically generates a pin assignment and an utilization report.

When the active implementation is prepared, the design is ready to be placed or analyzed for timing results. An encrypted Verilog netlist can also be generated for the prepared implementation for simulation. I/O pre-placement can also be performed when the design is prepared (see Pre-Placing a Design (see page 318)).

This flow step has the id `run_prepare` for Tcl commands.

Run Estimated Timing Analysis (Optional)

After **Run Prepare** has successfully completed on an implementation, the **Run Estimated Timing Analysis** step can be run. This step generates and writes a pre-place-and-route timing report file for the prepared design. The generated report is automatically displayed in the editor area upon successful completion. This step is run by default when **Run Flow** is executed.

This flow step has the id `report_timing_prepared` for Tcl commands.

Generate Pre-Placed Simulation Netlist (Optional)

After **Run Prepare** has successfully completed on an implementation, the **Generate Pre-Placed Simulation Netlist** step can be run. This step generates and writes an encrypted, pre-place-and-route Verilog netlist file from the prepared design. This netlist may be used to simulate the prepared design. This step is not run by default when **Run Flow** is executed.

This flow step has the id `write_netlist_prepared` for Tcl commands.

Place and Route Steps

Run Place

After **Run Prepare** has successfully completed on an implementation, the **Run Place** step must be run in order to place the design. If place and route has already been run on this implementation, this step may be skipped, and the place and route data from the previous run may be loaded by using the **File → Load Place and Route Data** menu option. When the design is successfully placed, the encrypted placement data is stored to disk and is ready to be loaded again later.

This flow step has the id `run_place` for Tcl commands.

Run Post-Placement Timing Analysis (Optional)

After **Run Place** has successfully completed on an implementation, the **Run Post-Placement Timing Analysis** step can be run. This step generates and writes a timing report file for the placed design, without requiring all final DRC checks to pass. The generated report is automatically displayed in the editor area upon successful completion. This step is not run by default when **Run Flow** is executed.

This flow step has the id `report_timing_placed` for Tcl commands.

Run Route

After **Run Place** has successfully completed on an implementation, the **Run Route** step must be run in order to route the design. If place and route has already been run on this implementation, this step may be skipped, and the place and route data from the previous run may be loaded by using the **File → Load Place and Route Data** menu option. When the design is successfully routed, the encrypted placement data is stored to disk and is ready to be loaded again later.

This flow step has the id `run_route` for Tcl commands.

Run Post-Route Timing Analysis (Optional)

After **Run Place** and **Run Route** have successfully completed on an implementation, the **Run Post-Route Timing Analysis** step can be run. This step generates and writes a timing report file for the placed and routed design, without requiring all final DRC checks to pass. The generated report is automatically displayed in the editor area upon successful completion. This step is not run by default when **Run Flow** is executed.

This flow step has the id `report_timing_routed` for Tcl commands.

Design Completion Steps

> **Caution!**
>
> All Flow Steps (see page 221) under the Design Completion category are skipped by default when the flow mode is set to **Evaluation**. See Flow Mode (see page 226) for more details.

Post-Process Design

After **Run Place** and **Run Route** have successfully completed (or a `.acxdb` file containing place and route data has been loaded) on an implementation, the **Post-Process Design** step must be run. This step post-processes the design by inserting Achronix-specific technology (such as reset, compensation block, and vmode insertion) that relies on final placement and routing information. This step should not affect timing results.

This flow step has the id `post_process` for Tcl commands.

Run Final DRC Checks

After **Post-Process Design** has successfully completed on an implementation, the **Run Final DRC Checks** step must be run. This step performs all final DRC checks in order to ensure that the bitstream, final timing, and final simulation netlist can be generated without errors. If a design fails final DRC checks, a **Post-Route** timing report can still be generated for experimental purposes. However, no bitstream may be generated to run the design on the hardware unless all final DRC checks pass.

This flow step has the id `final_drc_checks` for Tcl commands.

Run Sign-off Timing Analysis (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Run Sign-Off Timing Analysis** step can be run. This step generates and writes a final sign-off timing report file for the placed and routed design, after all final DRC checks have passed. The generated report is automatically displayed in the editor area upon successful completion. This step is run by default when **Run Flow** is executed.

This flow step has the id `report_timing_final` for Tcl commands.

Generate Final Reports (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Generate Final Reports** step can be run. This step generates various report files, including clocks, pins, power, etc. Implementation options are used to control which report files are generated. The generated report is automatically displayed in the editor area upon successful completion. This step is run by default when **Run Flow** is executed.

This flow step has the id `write_reports_final` for Tcl commands.

Generate Final Simulation Netlist (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Generate Final Simulation Netlist** step can be run. This step generates and writes an encrypted, post-place-and-route Verilog simulation netlist file from the final DRC-free design. This netlist may be used to simulate the post-place-and-route design. This step is not run by default when **Run Flow** is executed.

This flow step has the id `write_netlist_final` for Tcl commands.

FPGA Programming Steps

> **Caution!**
>
> The **Generate Bitstream** flow step fails if attempted in **Evaluation** flow mode. Additionally, all Flow Steps (see page 221) under the FPGA Programming category are skipped by default when the flow mode is set to **Evaluation**. See Flow Mode (see page 226) for more details.

Generate Bitstream

After a design is placed and routed, the **Generate Bitstream** step can be run. This step generates a bitstream (STAPL file) for the current implementation based on the settings in the Options view (see page 103) (see the Options settings for Bitstream Generation). This step is run by default when **Run Flow** is executed. The flow mode must be set to **Normal** before bitstream generation can complete successfully (while it typically produces much shorter flow runtimes, the **Evaluation** flow mode relaxes the DRCs too much to produce a reliable bitstream).

This flow step has the id `write_bitstream` for Tcl commands.

FPGA Download

After the bitstream is generated, it is ready for downloading to the FPGA via either the Achronix Bitporter 2 pod or an FTDI FT2232H device as specified under the Options view settings (see the Options settings for FPGA Download). This step is not run by default when **Run Flow** is executed.

This flow step has the id `fpga_download` for Tcl commands.

A bitstream can also be downloaded to the FPGA via the Download view (see page 51) (see Playing a STAPL File (Programming a Device) (see page 350))

For more details, refer to the *JTAG Configuration User Guide* (UG004).

Flow Status

Flow Steps (see page 221) each have a status associated with them. The current status of each flow step for the Active Project and Implementation (see page 221) can be seen in the GUI in the Flow View (see page 61).

Each step can be in one of the following flow states:

Table 128: *Flow State Icons*

State	Flow Category	Flow Step
Incomplete		
Running		
Complete		
Disabled		
Error		
Complete (but out of sync with source files)		

Be aware that changing or Configuring Implementation Options (see page 278) which affect a flow step can cause the status of a flow step to be reset back to Incomplete.

See the concepts for the Flow (see page 221) and Flow View (see page 61), as well as the task for Running the Flow (see page 280) for more details.

Flow Mode

The flow mode is managed as an Implementation (see page 215) Option, typically through the Options View (see page 103).

The chosen flow mode determines which DRCs are executed at different points in the Flow (see page 221), and in some configurations prohibit the final Flow Steps (see page 221) from executing successfully.

- **Evaluation** – this mode ignores non-fatal DRCs as long as possible, allows IO Virtualization, and ignores missing SDC constraints to get a post-route timing report quickly. This mode allows iterating more quickly during preliminary or early design stages.
- **Normal** – this flow mode enforces all DRC checks prior to generating a bitstream. Some checks are flagged as warnings early on in the flow to provide an opportunity to correct the problems (i.e., fixing the placement of I/Os). These same checks may change to report an error during final DRC checks. This mode should be used when developing a real design for a product, and enables bitstream generation.
- **Strict** – this mode is similar to **Normal** flow mode, but to reduce runtime, strictly enforces all DRC checks, erroring out as early in the flow as possible. This more restrictive mode should be used during the later, more mature design iterations.

When in **Evaluation** flow mode, the **Run Flow** and **Re-run Flow** actions in the Flow View (see page 61) stop after the **Place and Route** flow step category is completed. By default, the flows steps under **Design Completion** and **FPGA Programming** are not run unless the implementation is in **Normal** or **Strict** flow mode.

> **Note**
>
> The **Generate Bitstream** flow step fails if attempted while in **Evaluation** flow mode. Bitstream generation requires **Normal** or **Strict** flow mode, so that ACE may ensure all DRCs have passed.

Some examples of error checks in **Strict** flow mode, applicable to Speedcore devices, are as follows:

1. If any Speedcore boundary pins (IPIN/OPIN/CLK_IPIN/CLK_OPIN) are not explicitly instantiated in the user design RTL or PDC files, ACE errors out in the Run Prepare flow step.
2. If any Speedcore boundary pins (IPIN/OPIN/CLK_IPIN/CLK_OPIN) are not explicitly placed with "fixed" placement in the project PDC files, ACE errors out at the beginning of the Run Place flow step.

In other flow modes, these checks do not happen until Final DRC Checks prior to bitstream generation.

Reports

ACE generates a number of reports to provide information on how user designs are being handled in the selected Achronix device. These reports are meant to assist in making design decisions.

Each of the listed reports is generated in HTML format by default, and with the noted Tcl command, each report can optionally (unless otherwise noted) be generated in plain text format, or in CSV format for easy import into spreadsheet programs. As soon as the reports are generated, they are opened for viewing within the ACE Editor area.

Utilization Report

The Utilization Report shows a summary and details of the utilization of the device resources for the current design.

This report is automatically generated as part of the **Run Prepare** flow step (see page 221).

To generate this report manually, see the `report_utilization` Tcl command.

Pin Assignment Report

The Pin Assignment Report shows detailed information on each of the user design top-level ports, including placement and configuration details.

Typically, the report is automatically generated by the Flow (see page 221) at multiple times (as part of the **Run Prepare**, **Run Place**, and **Post-Process Design** flow steps (see page 221)).

To generate this report manually, see the `report_pins` Tcl command.

Clock Report

The Clock Report shows all clocks used in the design, their frequencies/periods, their relationships, and their constraints. Related information regarding device Clock Regions is also included in the report.

The report is automatically generated by the Flow (see page 221) at multiple stages of the flow.

To generate this report manually, see the `report_clocks` Tcl command.

Timing Report

The Timing Report provides details on how well the current design is meeting timing on the selected device.

Timing analysis can be performed at several stages in the Flow (see page 221), each stage generating a different report. If the design has not yet been routed, placement and/or routing are estimated.

This report is automatically generated by the flow during any of the **Run … Timing Analysis** Flow Steps (see page 221).

To generate this report manually, see the `run_timing_analysis` Tcl command.

If Timing Across All Temperature Corners (see page 246) is enabled, a separate timing report is generated for each temperature corner, with the file name of the report noting the corner being reported.

Report Content

The report contains a Summary section and a Details section.

The Summary section contains three tables. There is a table for Critical Setup (max) Timing Paths, one for Critical Hold (min) Timing Paths, and one for the resulting Clock Frequencies. Each summary section table contains a single row for each Clock/Group, showing the most critical path for that Clock/Group.

The Details section contains a configurable maximum number of critical setup paths and critical hold paths for each Clock /Group, and each of those critical paths includes a configurable maximum number of worst paths for the critical path endpoint.

The number of critical paths and worst paths are Implementation Options (see page 215) configured in the Options View (see page 103) under "Timing Analysis".

Routing Report

The Routing Report collects a number of routing-related statistics and any related errors into an easily readable report format.

This report is automatically generated by the Flow (see page 221) during the **Run Route** and **Post-Process Design** flow steps (see page 221).

To generate this report manually, see the `report_routing` Tcl command.

Partitions Report

The Partitions Report collects a number of partition-related statistics into an easily readable report format.

This report is automatically generated by the Flow (see page 221) (and opened in the GUI) during the **Run Prepare** flow step (see page 221) when Using Incremental Compilation (Partitions) (see page 363).

To generate this report manually, see the `report_partitions` Tcl command.

Power Dissipation Report

The Power Dissipation Report shows an estimate for the amount of power, in units of Watts, dissipated by the current design on the selected Achronix device under normal operating conditions. This report is typically generated after place-and-route, and therefore includes the effects of all transistor and wiring parasitics in the power calculation. This is in contrast to the Speedster7t Power Estimator and Speedcore Power Estimator tools, which are used early in the design process, prior to actual design activity.

Total power dissipation is reported so that thermal requirements can be estimated for the user design. Total power is also broken down into several different categories, such as power dissipated for each power rail, for each cell type, and instance power vs. interconnect power. Therefore the report can also be used for the design and sizing of the power supplies, and the selection of hard IP blocks such as DDR4 vs. GDDR6.

This User Guide provides an example Power Dissipation Report, with an explanation of each section of the report, as well as detailed information about how power dissipation is calculated, and how to generate the report.

Example Power Dissipation Report

The Power Dissipation Report consists of eight sections, each of which presents the power dissipation from a different perspective.

- Report Header

- Power Summary
- Hard IP Power
- Dynamic Power Per Cell
- Power Per Rail
- Core Dynamic Power Details
- Core Dynamic Instance Power Details
- Core Dynamic Interconnect Power Details section

Each section is described in more detail below.

Section: Report Header

This section, common to all ACE reports, provides information about how and when the report was generated. It can be seen that the design name is `pcie_gddr6_ddr4_vp_demo_top`, which is a VectorPath® card demonstration design that configures the PCIe, GDDR6, and DDR4 interfaces. The operating conditions assumed for this report can also be seen: C2 speed grade, 0.85 volts, and 0 degrees Celsius.

Power Dissipation Report

ACE -- Achronix CAD Environment -- Version 8.8.2 -- Build 369189 -- Date 2022-11-02 14:17
Design: pcie_gddr6_ddr4_vp_demo_top - impl_1 - pcie_gddr6_ddr4_vp_demo_top
Device: AC7t1500ES0
Generated on Wed Nov 16 18:00:03 PST 2022
Host: cad53.achronix.local

Disclaimer

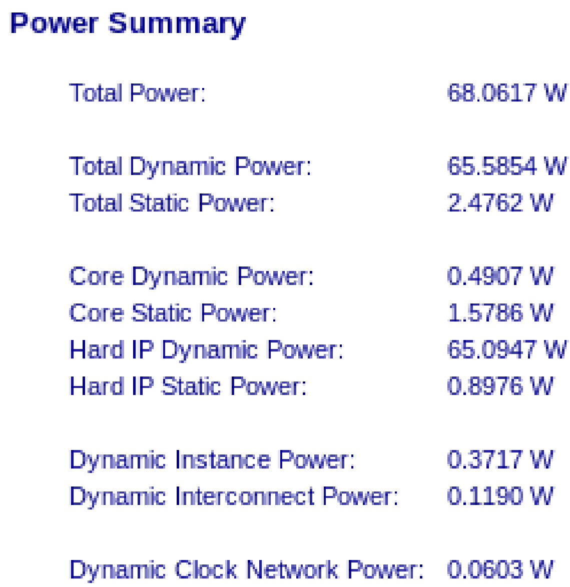
This is an estimation, based on the design characteristics and frequencies specified below. Power consumption is reported as worst case across all operating modes, including bitstream programming and user mode. Actual power results may vary from the estimates.

Operating conditions: C2, 0 deg. C, 0.85 V (Core), Fast Corner

Figure 106: ***Example of Power Dissipation Report Header Section***

Section: Power Summary

This section reports the total power dissipation, which is about 68 Watts, and then breaks that total down into categories such as static vs. dynamic power, core logic vs. hard IPs, and instances vs. interconnect.



Power Summary

Total Power:	68.0617 W
Total Dynamic Power:	65.5854 W
Total Static Power:	2.4762 W
Core Dynamic Power:	0.4907 W
Core Static Power:	1.5786 W
Hard IP Dynamic Power:	65.0947 W
Hard IP Static Power:	0.8976 W
Dynamic Instance Power:	0.3717 W
Dynamic Interconnect Power:	0.1190 W
Dynamic Clock Network Power:	0.0603 W

Figure 107: ***Example of Power Dissipation Report Power Summary Section***

Section: Hard IP Power

This section reports details of the power contributed by each hard IP block in the design. The example design contains many of the available hard IP blocks, so the report contains many rows. For example, it can be seen that more than half of the total power dissipation in the design, which contains very little core logic, is used by the GDDR6 controller.

In order to include the power contribution of hard IP blocks, those IPs must have been configured in the IP Configuration editors (see page 306), and also included the associated IP Configuration Files (see page 218) in their ACE project.

Hard IP Power

CLKIO_NE:	0.4635 W
CLKIO_NW:	0.5251 W
CLKIO_SE:	0.4812 W
CLKIO_SW:	0.4507 W
DDR4:	2.7135 W
Ethernet:	0.0096 W
GDDR6:	43.1243 W
GPIO_N_B0:	0.3100 W
GPIO_N_B1:	0.3100 W
GPIO_N_B2:	0.2348 W
GPIO_S_B0:	0.3034 W
GPIO_S_B1:	0.1381 W
GPIO_S_B2:	0.2283 W
NoC:	15.2885 W
PCIe:	0.2099 W
PLL:	0.2922 W
SERDES_LN_0:	0.0038 W
SERDES_LN_1:	0.0038 W
SERDES_LN_10:	0.0038 W
SERDES_LN_11:	0.0038 W
SERDES_LN_12:	0.0038 W
SERDES_LN_13:	0.0038 W
SERDES_LN_14:	0.0038 W
SERDES_LN_15:	0.0038 W
SERDES_LN_2:	0.0038 W
SERDES_LN_3:	0.0038 W
SERDES_LN_4:	0.0038 W
SERDES_LN_5:	0.0038 W
SERDES_LN_6:	0.0038 W
SERDES_LN_7:	0.0038 W
SERDES_LN_8:	0.0038 W
SERDES_LN_9:	0.0038 W
SoC:	0.8493 W

Figure 108: ***Example of Power Dissipation Report Hard IP Power Section***

Section: Dynamic Power Per Cell

This section reports details of the dynamic power contributed by instances of each cell type, such as Reconfigurable Logic Blocks (LUT6s, DFFs, and ALUs), Block RAMs, Network Access Points, etc. This design contains very little programmable core logic, so these numbers are quite small relative to the Hard IPs.

Dynamic Power Per Cell

RLB Power:	0.0041 W
BRAM Power:	0.0017 W
LRAM Power:	0.0013 W
MLP Power:	0.0000 W
NAP Power:	0.3646 W
Hard IP Power:	65.9923 W
Other Power:	0.0000 W

Figure 109: ***Example of Power Dissipation Report Dynamic Power Per Cell Section***

Section: Power Per Rail

This section reports the power dissipation for each power rail in the design (rows), breaking it down into categories (columns) such as dynamic vs. static power, and core logic vs. hard IPs. For example, the VDDL power rail represents the programmable logic core of the design, and it contributes the majority of the static power. The rest of the power rails are used by the hard IPs.

Power Per Rail

Power Rail	Total Power (W)	Total Dynamic Power (W)	Total Static Power (W)	Core Dynamic Power (W)	Core Static Power (W)	Hard IP Dynamic Power (W)	Hard IP Static Power (W)
CLKIO_NE_VDDIO	0.4124	0.4124	0.0000	0.0000	0.0000	0.4124	0.0000
CLKIO_NW_VDDIO	0.4124	0.4124	0.0000	0.0000	0.0000	0.4124	0.0000
CLKIO_SE_VDDIO	0.4124	0.4124	0.0000	0.0000	0.0000	0.4124	0.0000
CLKIO_SW_VDDIO	0.4124	0.4124	0.0000	0.0000	0.0000	0.4124	0.0000
DDR4_S0_VAA	0.0151	0.0130	0.0021	0.0000	0.0000	0.0130	0.0021
DDR4_S0_VDDQ	0.8913	0.8520	0.0393	0.0000	0.0000	0.8520	0.0393
FCU_VDDIO	0.8452	0.8452	0.0000	0.0000	0.0000	0.8452	0.0000
GCG_NE_PLL_VDDA	0.0042	0.0042	0.0000	0.0000	0.0000	0.0042	0.0000
GCG_NW_PLL_VDDA	0.0042	0.0042	0.0000	0.0000	0.0000	0.0042	0.0000
GCG_SE_PLL_VDDA	0.0042	0.0042	0.0000	0.0000	0.0000	0.0042	0.0000
GCG_SW_PLL_VDDA	0.0166	0.0166	0.0000	0.0000	0.0000	0.0166	0.0000
GDDR6_E_VDDA	8.4480	8.4480	0.0000	0.0000	0.0000	8.4480	0.0000
GDDR6_E_VDDIO	4.2780	4.2780	0.0000	0.0000	0.0000	4.2780	0.0000
GDDR6_E_VDDP	4.6350	4.6350	0.0000	0.0000	0.0000	4.6350	0.0000
GDDR6_E_VDDR	4.2012	4.1837	0.0174	0.0000	0.0000	4.1837	0.0174
GDDR6_W_VDDA	8.4480	8.4480	0.0000	0.0000	0.0000	8.4480	0.0000
GDDR6_W_VDDIO	4.2780	4.2780	0.0000	0.0000	0.0000	4.2780	0.0000
GDDR6_W_VDDP	4.6350	4.6350	0.0000	0.0000	0.0000	4.6350	0.0000
GDDR6_W_VDDR	4.2012	4.1837	0.0174	0.0000	0.0000	4.1837	0.0174
GPIO_N0_VDDIO	0.4883	0.4883	0.0000	0.0000	0.0000	0.4883	0.0000
GPIO_S0_VDDIO	0.3231	0.3231	0.0000	0.0000	0.0000	0.3231	0.0000
SRDS_N_PA_VDDH	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SRDS_N_PA_VDDL	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TS_VDDA	0.0004	0.0004	0.0000	0.0000	0.0000	0.0004	0.0000
VCC	18.6260	17.8047	0.8214	0.0000	0.0000	17.8047	0.8214
VDDL	2.0693	0.4907	1.5786	0.4907	1.5786	0.0000	0.0000

Figure 110: ***Example of Power Dissipation Report Power Per Rail Section***

Section: Core Dynamic Power Details

This section reports the programmable core dynamic power by power rail and clock domain (rows), breaking it down into categories (columns) for instance power, interconnect power, and clock network power.

Core Dynamic Power Details

Power Rail	Clock	Frequency (MHz)	Total Dynamic Power (W)	Instance Power (W)	Interconnect Power (W)	Clock Network Power (W)
VDDL	i_nap_clk	400.00	0.1594	0.1158	0.0200	0.0236
VDDL	i_reg_clk	200.00	0.2849	0.2258	0.0284	0.0307
VDDL	i_adm_clk	100.00	0.0459	0.0301	0.0101	0.0056
VDDL	tck	25.00	0.0005	0.0000	0.0001	0.0004
VDDL	Combinational	-----	0.0001	0.0000	0.0000	0.0000

Figure 111: *Example of Power Dissipation Report Core Dynamic Power Details Section*

Section: Per-Tile Core Dynamic Instance Power Details

This section reports the programmable core dynamic instance power by power rail and clock domain (rows), breaking it down into categories (columns) for each type of logic tile, such as Reconfigurable Logic Block tiles (RLBs), MLP tiles, BRAM tiles (not shown), and 2D NoC tiles.

Per-Tile Core Dynamic Instance Power Details

Power Rail	Clock	Frequency (MHz)	Total Instance Power (W)	Clock Power (W)	IO Pin Power (W)	MLP Power (W)	NOC Power (W)	RLB Power (W)
VDDL	i_nap_clk	400.00	0.1158	0.0000	0.0000	0.0013	0.1133	0.0012
VDDL	i_reg_clk	200.00	0.2258	0.0000	0.0000	0.0008	0.2230	0.0019
VDDL	i_adm_clk	100.00	0.0301	0.0000	0.0000	0.0008	0.0283	0.0010
VDDL	tck	25.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
VDDL	Combinational	-----	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 112: *Example of Power Dissipation Report Per-Tile Core Dynamic Instance Power Details Section*

Section: Per-Tile Core Dynamic Interconnect Power Details

This section reports the programmable core dynamic interconnect power by power rail and clock domain (rows), breaking it down into categories (columns) for each type of logic tile, such as Reconfigurable Logic Block tiles (RLBs), MLP tiles, BRAM tiles (not shown), and 2D NoC tiles. Interconnect power includes the actual routing wires, as well as the statically programmable FPGA routing MUXes used to connect different wire segments to form a complete route.

Per-Tile Core Dynamic Interconnect Power Details

Power Rail	Clock	Frequency (MHz)	Total Interconnect Power (W)	Clock Power (W)	IO Pin Power (W)	MLP Power (W)	NOC Power (W)	RLB Power (W)
VDDL	i_nap_clk	400.00	0.0200	0.0136	0.0000	0.0013	0.0010	0.0276
VDDL	i_reg_clk	200.00	0.0284	0.0189	0.0000	0.0042	0.0022	0.0338
VDDL	i_adm_clk	100.00	0.0101	0.0021	0.0000	0.0009	0.0003	0.0125
VDDL	tck	25.00	0.0001	0.0003	0.0000	0.0000	0.0000	0.0002
VDDL	Combinational	-----	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 113: *Example of Power Dissipation Report Per-Tile Core Dynamic Interconnect Power Details Section*

Power Calculation Methodology

Power dissipation is calculated using the following basic equation:

$$power_{total} = \sum_{nets}(energy_{net} \times frequency_{net} \times activity_{net})$$

Here total power is summed up over the individual contribution of every net in the design, including nets internal to logic gates and hard IP blocks. Power for each net is calculated as the product of the total `energy` dissipated in each zero-to-one or one-to-zero transition, measured in units of joules, which is derived from extracted layout data and present in the technology library being used; the target `frequency` of that net clock domain, measured in units of 1/seconds; and a unitless quantity called `activity`, which is defined as the average number of transitions *per clock cycle*.

`Activity` can be larger than one if a net experiences a lot of glitching. The activity factor for clock nets is, by definition, 2.0, since there are two transitions per cycle. ACE typically uses a default value of `0.125` for the activity value of non-clock nets. This value reflects an assumption that non-clock nets switch, on average, about once every eight clock cycles. This has been shown to be a reasonable assumption across a wide variety of designs, but it is, of course, not always correct. In order to produce the most accurate power report possible, per-net activity values can be specified in a Switching Activity Interchange Format (SAIF) file. SAIF files are typically produced by a gate-level logic simulator. However, if necessary, the file can be written by hand. The accuracy of the activity values in a SAIF file depend on the quality of the simulation testbench, and are only reflective of the operating mode(s) exercised in that testbench. See the section How to Generate a Simulation-Driven Power Dissipation Report (see page 234) below for more information.

The Power Dissipation Report can optionally include the contribution of hard IP blocks, such as the PCIe controller and the 2D Network on Chip. In order to include those contributions (which can often dominate the contribution of the programmable logic), those hard IP blocks must have been configured in the IP Configuration editors (see page 306), and also have included the associated IP Configuration Files (see page 218) in the ACE project.

How to Generate the Power Dissipation Report

This report is generated automatically when running the ACE Flow (see page 221) as part of the `write_reports_final` flow step. However, it is not generated by default. The `report_power` implementation option (see page 278) must be set to `"1"` in the project before running the `write_reports_final` flow step.

This report can also be generated on-demand by restoring the routed implementation (see page 277) and then calling the `report_power` Tcl command.

As power dissipation varies across different process corners and operating conditions, a separate power dissipation report is generated for each temperature corner, with the file name of the report noting the corner being reported. See the page Timing Across All Temperature Corners (see page 246) for more information.

In order to include the power contribution of hard IP blocks, those IPs must have been configured in the IP Configuration editors (see page 306), and also have included the associated IP Configuration Files (see page 218) in the ACE project.

How to Generate a Simulation-Driven Power Dissipation Report

As discussed above in the Power Calculation Methodology section, per-net `activity` values can be specified in a text file called a Switching Activity Interchange Format (SAIF) file. SAIF files are typically produced by a gate-level logic simulator such as Synopsys VCS, Cadence Incisive, Siemens QuestaSim, or Aldec Riviera. This section covers the format of a SAIF file, how to use a SAIF file in ACE, and how to generate a SAIF file in the VCS and QuestaSim simulators. The documentation should be consulted for the simulator of choice for definitive instructions.

Format of a SAIF File

SAIF files are written in a clear-text format defined in IEEE Standard 1801-2009 (see https://ieeexplore.ieee.org/document/8686430). Therefore, if desired, small SAIF files can be generated by hand or with a script. The standard defines both *backward* and *forward* SAIF file formats. Here we are using a *backward* SAIF file which is intended for back-annotation into EDA tools. Below is an example of a complete SAIF file that is accepted by ACE.

Sample SAIF File

```
(SAIFILE
   (SAIFVERSION "2.0")
   (DIRECTION "backward")
   (DURATION 5950100.00)
   (INSTANCE tb_test1
      (INSTANCE DUT
         (NET
            (ina\[0\]
               (T0 3750100) (T1 2200000) (TX 0)
               (TC 22) (IG 0)
            )
            (clk
               (T0 3000100) (T1 2950000) (TX 0)
               (TC 118) (IG 0)
            )
         )
      )
   )
)
```

The file must:

1. Begin with the `SAIFILE` keyword.
2. Give the version number.
3. Specify that it is a backward SAIF file.
4. Give the duration of the simulation as the number of verilog timesteps.

Most important are the `INSTANCE` and `NET` sections of the file. It can be seen that this design has two levels of hierarchy. The instance `tb_test1` is the testbench containing behavioral verilog that applies test vectors to the device under test. The instance `DUT` is the Device Under Test, which is an instance of the top level design being compiled in ACE. The design contains two nets, an input net, `ina[0]`, and a clock net, `clk`. Square brackets and other reserved characters must be escaped with "\" in the SAIF file.

Under each net are five values, `T0`, `T1`, `TX`, `TC`, and `IG`. The `T0`, `T1`, and `TX` values are the number of timesteps that the net spent at the logic values 0, 1, and X respectively. The `IG` value is the number of inertial glitch values — for details see the documentation. The only important value for ACE is the `TC` value, which is the total number of zero-to-one and one-to-zero transitions over the duration of the simulation. Since the clock net had 118 transitions during simulation, and the `ina[0]` net had 22 transitions, ACE calculates the activity value, which is defined as the average number of transitions per clock cycle, as `22/118 = 0.1864` (not very different from the default activity value of 0.125).

How to Use a SAIF File in ACE

To use a SAIF file, two parameters must be specified to ACE:

1. The pathname to the SAIF file.

2. The name of the top-level instance in the simulation testbench.

The Tcl interpreter normalizes the pathname, so it can be an absolute or relative path, a direct pathname or a symbolic link, and in Windows or Linux format. The name of the top-level instance in the SAIF file must be specified because ACE only knows the name of the top-level *module* in the RTL. Since the testbench is not being compiled, ACE does know the name of the *instance* of that module in the behavioral simulation testbench module. The default SAIF file pathname is the empty string (no file specified). The default top-level instance name is "DUT", as in this example.

When generating the Power Dissipation Report during the normal ACE flow, specify the pathname to the SAIF file in the ACE project file using the `report_power_saif_file` implementation option, and the name of the top-level instance in the `report_power_saif_top_level` implementation option:

```
set_impl_option -project test1 -impl impl_1 report_power_saif_file "../vcs-routed/sim_output.saif"
set_impl_option -project test1 -impl impl_1 report_power_saif_top_level "DUT"
```

When generating the Power Dissipation Report from the ACE command line, simply call the `report_power` command with the following two options:

```
report_power -saif_file ../vcs-routed/sim_output.saif -saif_top_level DUT
```

How to Generate a SAIF File Using Synopsys VCS

In this section, for convenience, we provide instructions for how to generate a SAIF file using Synopsys VCS. However, note that VCS behavior may change at any time, and these instructions may be incorrect or incomplete. More than one method of generating a SAIF file may be supported. Consult your simulator documentation for definitive instructions.

There are four steps required:

1. Generate a Final Simulation Netlist (a post-routing simulation netlist that matches the netlist used by the power report) using ACE. The Generate Final Simulation Netlist (see page 224) flow step is an optional flow sub-step that runs under the Design Completion flow step, so that sub-step must be enabled before running the ACE flow.
2. From within the verilog testbench, several additional system functions must be called to specify which part of the design to be observed, and to start and stop the recording of data. The following code can be used as a rough template for a simulation testbench:

```
initial begin
    $set_gate_level_monitoring("on");
    $set_toggle_region(mixedHdlScope);
    // initialization of Verilog signals, and then:
    $toggle_start;
    // testbench
    $toggle_stop;
    $toggle_report("sim_output.saif", $timeUnit, mixedHdlScope);
end
```

3. Call the VCS compiler to generate the `simv` executable. To generate a SAIF file, the `-debug_access+pp` flag must be added:

```
vcs <input_files> <testbench> -debug_access+pp
```

4. Call the simulation executable. To generate a SAIF file, the `-lcs` (limited customer access) flag must be added:

```
simv -lcs
```

How to Generate a SAIF File Using Siemens QuestaSim

In this section, for convenience, we provide instructions for how to generate a SAIF file using Siemens QuestaSim. However, note that QuestaSim behavior may change at any time, and these instructions may be incorrect or incomplete. More than one method of generating a SAIF file may be supported. Consult your simulator documentation for definitive instructions.

There are seven steps required:

1. Generate a Final Simulation Netlist (a post-routing simulation netlist that matches the netlist used by the power report) using ACE. The Generate Final Simulation Netlist (see page 224) flow step is an optional flow sub-step that runs under the Design Completion flow step, so that sub-step must be enabled before running the ACE flow.
2. Launch QuestaSim in interactive mode, while specifying the additional option `-voptargs="+acc"`. This option prevents QuestSim from compressing the netlist and throwing away nodes that must be reported.

```
vsim -voptargs="+acc"
```

3. After the simulator returns to the prompt, run the following command to request the simulator to stop rather than exiting after the simulation completes:

```
onfinish stop
```

4. Specify the scope of the data to be collected, which should include all nets in the DUT. Using the example SAIF file above, this would be as follows:

```
power add -in -inout -internal -out tb_test1.DUT.*
```

5. Run the simulation:

```
run -all
```

6. After the simulation completes, request the simulator to write out the SAIF file:

```
power report -al -bsaif sim_output.saif
```

7. Quit QuestaSim:

```
quit
```

Design Statistics Report

The Design Statistics Report is meant to show various statistics about the current design.

Presently, the only statistics being reported are a histogram showing LUT Function usage counts. This information is primarily meant as a tool for Achronix Technical Support to assist ACE users unable to share their full design. It allows Achronix to better understand the nature of the design and thus provide advice on QoR improvements.

This report is not automatically generated by the Flow (see page 221), but can be generated manually with the `report_design_stats Tcl command.`

Multiprocess Summary Report

The Multiprocess Summary Report provides a comparative summary of the achieved frequencies and worst-case slacks (if the "Run Post-Route Timing Analysis" or "Run Sign-off Timing Analysis" Flow Steps (see page 221) are enabled), as well as process execution times, for selected Implementations (see page 215) of a single Project when executed from the Multiprocess View (see page 83). This report is automatically shown (and refreshed) in the ACE Editor Area as new results become available during Multiprocess execution.

The Multiprocess Summary Report is generated/updated multiple times during a Multiprocess execution as new data becomes available from the executing Implementations (see page 215). While the Multiprocess flow is still executing, the report contains incomplete results — rows containing incomplete data are marked as such. Similarly, if errors are encountered during flow execution for one of the selected Implementations (see page 215), the row(s) of data for that implementation are marked accordingly.

Because the Multiprocess Summary Report summarizes the results of multiple Implementations (see page 215) (unlike the other reports, which are generated for a single implementation), the HTML file containing the report is not placed in any implementation `report` subfolder. Instead, this report is placed in the directory containing the `.acxprj` ACE Project File (see page 216), and is named `multiprocess_summary.html`.

Tip

If closed, the Multiprocess Summary Report can be re-opened at any time.

To re-open the Multiprocess Summary Report for the active project, select the ( ) **Open Multiprocess Report** button in the upper-right of Multiprocess view, or from the context menu when right-clicking the project in the Projects View (see page 125).

This report is only available in HTML format. There is no Tcl command available to generate this report manually.

For more information on how this report is used, see Running Multiple Flows in Parallel (see page 282) and Attempting Likely Optimizations Using Option Sets (see page 352).

Timing Results Summary Section

The Timing Results Summary section of the report is only generated if either the "Run Post-Route Timing Analysis" or "Run Sign-off Timing Analysis" Flow Steps (see page 221) are executed during the Multiprocess Flow.

Caution!

Any desired optional flow steps in the Flow View (see page 61) must be enabled before Multiprocess execution is started.

The completed implementations in this section are sorted in approximate QoR order by their timing results. The implementation with the best results is at the top (also see the `get_best_multiprocess_impl` Tcl command.)

While an implementation is waiting to be executed or is currently executing, the Clock Domain column of the report contains a message to indicate the implementation execution status.

When an implementation has completed execution, the timing results for that implementation are populated in the Report. By default, each implementation provides timing results (Upper-Limit Frequency, Worst Setup Slack, and Worst Hold Slack) for a single PVT combination (named in a column group header). However, if the "Report all temperature corners" implementation option is enabled (see the Options View (see page 103)), multiple PVT combinations are reported, each under its own column group header. The active PVT combination (the specific combination chosen for an implementation, which would otherwise be the only one reported) are shown in bold to differentiate it from other PVT combinations shown for that implementation.

The Upper-Limit Frequency, Worst Setup Slack, and Worst Hold Slack cells are independently color-coded to indicate whether the timing constraints were met for each of the defined clock domain/PVT combinations for that implementation. If the timing constraints were met for all clocks and all reported PVT combinations in an implementation, the Implementation Name cell is also color coded green to indicate success. If errors are encountered during flow execution, the appropriate Implementation Name cells are colored red, and " ***(Flow Error)*** " appears in that row.

Hyperlinks to each detailed Timing Report (see page 227) are made available under the first clock domain Upper-Limit Frequency column for each PVT (one is created for each implementation for each reported PVT).

In some cases, some implementations provide Sign-Off timing results but other implementations do not. This difference can happen most often when some implementations are in Evaluation Flow Mode (see page 226) while others are not. It may also happen when the multi-process flow is cancelled, or in rare flow error cases. When this mix of timing results from differing flow steps occurs, the column headings indicate that the report contains Sign-Off data. All earlier Post-Route timing results are footnoted to indicate that they are not showing Sign-Off data.

Runtime Results Summary Section

This section of the report is always generated and indicates the total flow execution time and peak memory utilization for each implementation. The flow runtimes reported are wall-clock runtimes (not CPU times) harvested from the run logs. Rows in the table indicate whether an implementation flow execution is incomplete (running or still waiting to run), or has encountered errors.

Many factors affect runtimes:

- Selected Implementation Options (see page 215) for an implementation (and thus the Option Sets (see page 215)) can have a significant impact.
- The available processors, available memory, and total load on the workstation executing ACE have a significant impact.
  - On multi-user workstations, workstation load may vary widely over the multiprocess execution period.
  - Even on single-user workstations, if using a **Parallel Queue Count** greater than one, be aware that the last-executed implementation(s) likely can be executing (at least partially) with a reduced machine load compared to the first-executed implementations. As the parallel execution queue is emptied, new implementation processes are not started, thus fewer processes are executing, meaning that the last-executed implementations have the lowest contention for processing cores, caches, memory, I/O, etc.
- Additionally, when using external cloud/grid/batch Job Submission systems:
  - The reported times do not include the time spent waiting in the external system queue(s). The runtimes reported only include time spent actually running the ACE flow.
  - The processor speeds and system loads may vary widely for the nodes running each job, making runtime comparisons difficult (at best).
  - If meaningful runtime comparisons are required, extra care must be taken to ensure that each node system load and hardware is identical for all jobs for the duration of the multiprocess session.

Caution!

Do not compare runtimes unless workstation hardware and system loads are identical for all implementations. Because of all of the limitations listed above, the reported implementation flow execution wall-clock times are intended to be used only as general guidelines, not as benchmarking times.

Implementation Options Report

The Implementation Options Report provides information about available options for the currently active implementation. This report is not automatically generated by the Flow (see page 221), but can be generated on demand using the `report_impl_options` Tcl command.

By default, the report only displays the names, descriptions and default values of the most commonly used implementation options (meaning the subset which is shown within the Options View (see page 103)). In addition, by default, the report displays the options applicable to the target device of the currently active implementation.

The `-project` and `-impl` arguments can be used to show the options applicable to a different project implementation.

The `-show_values` argument can be used to include the current implementation value of each option in the report.

The `-show_all` argument can be used to include all possible options for the specified implementation in the report, instead of only the most-commonly-used subset shown within the Options View (see page 103).

Note

The values of hidden options (those which are not shown in the GUI Options View) should only be altered after guidance from Achronix support.

Advanced Concepts

The following are advanced concepts intended primarily for extremely experienced users, or users being actively guided by Achronix FAEs.

ACE Verilog Attributes

This page lists various attributes that can be applied to instances, nets, pin, ports, or other objects in the ACE data model. Each attribute is listed with the type of object or objects to which it may be applied, and a description of how it effects the ACE synthesis, placement, and routing flow.

locked

Applies to nets only.

Nets marked with this attribute are not unrouted by the `run_unroute` command, or when the `run_route` flow step is re-run on a routed design.

fanout_limit

Applies to nets only.

This is a dual-use property and can be applied globally and also individually to nets.

The global limit is specified with the `fanout_limit` impl option.

When applied to an individual net, this attribute overrides the global limit. Net drivers are cloned, or buffer trees inserted, to keep each (non clock/reset) net fanout under this limit. Applies only when the fanout_control impl option is enabled.

> **Note**
>
> ACE never inserts more fanout buffers than the maximum specified by the `max_buffer_limit` impl_option.

In order to find the correct net name, if the name of the driving register is known, the following Tcl command can be used:

```
set_property fanout_limit 50 [ find {*} -nets -filter @driving_pin=[get_pins *<source reg>*q] ]
```

must_keep

Applies to instances or nets.

The instance or net cannot be deleted by any of the netlist optimization flow commands.

> **Note**
>
> If the instance or net is logically redundant, it might be left dangling with no input and no output connections.

do_not_rewire

Applies to instances, nets, pins.

Rewiring permutes input pins among equivalent nets (i.e., nets in a tree of fanout-control buffers) to minimize wirelength and improve timing. This attribute disables rewiring for a given instance, net, or pin. Some rewiring can happen during `run_prepare`, but the majority of changes are made after `run_place` and `run_route`. Subject to the `prepnr_rewire` and `postpnr_rewire` implementation options.

do_not_clone

Applies to instances only.

Prevents a given instance from being cloned during fanout control optimization. However, fanout buffers may still be inserted for nets above the fanout_limit.

do_not_cluster

Applies to instances only.

Prevents an instance from being clustered during structural or timing-driven clustering. Structural clustering (enabled with the `structural_clustering_mode` implemenation option) creates clusters from groups of LUTs and Flops when certain pre-defined structures are encountered. Timing-driven clustering (enabled by the `timing_driven_clustering` implementation option) creates larger clusters from LUT-to-LUT and ALU-to-LUT connections to keep timing-critical net routing short. Instances in a cluster are placed together.

clock_type

Applies to nets or driver (output) pins.

Normally, this property is set on a net or driver (output) pin using the `set_clock_type` TCL command in the SDC constraints, but it can also be specified using this attribute. Applies globally to all target pins driven by that net or pin. For more information see the documentation for the `set_clock_type` Tcl command. The attribute must be a comma-separated list of the following strings: {"boundary", "trunk", "direct_trunk", "minitrunk" (Speedcore only), "blocked", "data_region", "data_center", "data_local", "branch_fast", "branch_nominal", "none", "low_jitter", "local", "global", "immediate"}.

> **Note**
>
> Not all attribute combinations make sense.

local_clock_type

Applies to pins only.

Has all of the same properties as the clock_type attribute above, but use this attribute when the type is being specified for a specific target (input) pin, and the routing type needs to be different than the global value specified for the driving net /pin. Useful for certain kinds of data-generated clocks: parts of the clock that feed back should be "data" but the rest should be "data_region" or "data_center".

syn_useioff

Applies to ports, nets, I/O pad/pin instances, or on flop flop instances.

Depending on the value of the `push_flops_into_pads` implementation option, the presence of this property causes boundary flop flop instances to be pushed into the attached input or output pad/pin instance. For more information see the Automatic Flop Pushing into I/O Pads (see page 417) section.

ace_virtualize

Applies to ports only.

Allows specifying which ports and port busses are virtualized when ACE is run in evaluation flow mode, and when the design contains more top level ports than available I/O pads/pins. For more information see the Working with Virtual I/O (see page 425) section.

ace_virtualize_clock_port

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows specifying the top-level port name to be connected to the clock input of the new serialization flop instances. Cannot be used with the `ace_virtualize_clock_net` attribute (they are mutually exclusive). For more information see the Working with Virtual I/O (see page 425) section.

ace_virtualize_clock_net

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows specifying the top-level net name to be connected to the clock input of the new serialization flop instances. Cannot be used with the `ace_virtualize_clock_port` attribute (they are mutually exclusive). For more information see the Working with Virtual I/O (see page 425) section.

async_capture

Applies to a DFF "d" input pin only.

Suppresses setup-and-hold checks at the input pin of a DFF instance during Standard Delay Format (SDF) export for the Timing Annotated Gate Level Simulation flow. This attribute is used on the capture flop of user-supplied clock domain crossing synchronizer macros. For more information, refer to the relevant IP Component Library User Guide for that product family.

location

Applies to instances only.

A string attribute giving the site name on which the instance is to be pre-placed. Equivalent to the "`set_placement -fixed`" PDC constraint.

Clock Regions

Device fabrics deal with numerous clocks. Due to physical routing limitations, only a finite number of clocks can be routed to each individual site within the fabric. To keep the placement/routing problem space manageable for the most complex designs, a fabric is divided up into Clock Regions of a relatively coarse granularity, where each Clock Region as a whole allows a finite number of clocks to be routed within that clock region.

Each fabric is divided up into a number of Clock Regions of roughly similar area. The exact numbers of clock regions, the dimensions of each region, the number and types of sites within the region, the allowed sources of the clocks routed to each region, and the differences (like skew) of clock behavior between clock regions all are specified by the chosen target device. A subset of this information is presented in the Clock Regions View (see page 36). See the technical specification of the target device for complete details.

For designs with an extremely large number of clocks, the use of Placement Regions and Placement Region Constraints (see page 354) may be necessary to guide placement decisions regarding Clock Regions. This should be discussed thoroughly with an Achronix FAE first, as improper use of Placement Region Constraints can lower QOR or even cause Placement or Routing to become unsolvable.

Instance States

An individual Instance can have a variety of states simultaneously in ACE, and only the highest priority state is used to color the Instance in the Floorplanner View (see page 53). The states are listed below in the default render priority order. Higher priority states (earlier in the table) take precedence over lower priority states (if an instance has Fixed Placement and is also Selected, the Selection color has priority, and the Selection color is used to paint the instance).

Additionally, several of the states have associated icons, which normally are displayed alongside the instance when the instance appears in tables and lists, as in the Search View (see page 132), Selection View (see page 136), and Netlist Browser View (see page 89). Similar to the colors, the highest priority icon is used. Thus, an instance that is both Fixed and Locked uses the Fixed icon.

Table 129: *Instance States By Priority Order*

Priority	State	Default Color	Icon	Description
1	Selected	bright green	–	This instance has been added to the ACE Selection Set, typically either by using the Selection View (see page 136) or the Tcl select (see page 600) command. For more information, see Selecting Floorplanner Objects (see page 313).
2	Highlighted	(user-defined)	–	An instance chosen to Highlight, typically by using the Highlight Instance commands in one of the views within the Floorplanner Perspective, or by the `highlight` Tcl command. See Highlighting Objects in the Floorplanner View (see page 315) for more information.

Priority	State	Default Color	Icon	Description
3	Overassigned Site	bright red	–	An instance that shares a site assignment with another instance. Since a site can only legally contain a single instance, this is an error state.
4	Fixed Placement	bright yellow		An instance whose site assignment has been marked as "fixed". As long as an instance is defined with hard fixed placement, ACE does not change the site assignment for that instance during the Placement phase of PnR. For more information, see Placing an Object (see page 318).
5	Locked Placement	dark yellow		An instance that is a member of a Locked Partition that has remained unchanged since the prior incremental compilation. ACE does not change the site assignment for that instance during the Placement phase of PnR. For more information, see Using Incremental Compilation (Partitions) (see page 363).
6	Default (Soft) Placement	dark grey		A placed instance with no other specially defined state is displayed in this manner.
7	Unplaced	–		An instance without a current site assignment (placement).

Note

The colors mentioned are the default colors. These colors may be modified on the Floorplanner View Colors and Layers Preference Page (see page 190). On that same preference page, it is possible to toggle whether some states are shown at all, and to partly alter the render priority of some states.

Filter Properties

Several Tcl commands `[ find, filter ]` allow Tcl command-line filtering of object lists. Additionally, the Search Filter Builder Dialog (see page 174) performs a similar function for the Search View (see page 132) in the ACE GUI.

The allowed filtering properties, operators, and values (where applicable) are as follows:

Table 130: *Supported Filter Properties*

Property Name	Operators	Values	Description
@attribute	=	*property*:*value*	The @attribute filter allows filtering instances, nets, and ports based on verilog attribute/defparam values. Values of the @attribute filter must be a property-value pair separated by a semicolon (i.e., prop:value).
@clock_domain [(1)]	=	*clockDomainName*	The @clock_domain filter allows filtering instances, nets, paths, and pins based on clock domain name.
@direction	=	in, out, inout	The @direction filter allows filtering ports and pins based on direction.
@driver_type	=	(device-dependent)	The @driver_type filter allows filtering nets and pins based on the type (cell name) of the driving instance.

Property Name	Operators	Values	Description
@driving_net	=	*netName*	The @driving_net filter allows filtering instances based on a net name that is driving them.
@driving_pin	=	*pinName*	The @driving_pin filter allows filtering instances and nets based on the name of a pin that is driving them.
@fanout	=, <, >	(integers > 0)	The @fanout filter allows filtering nets and pins based on number of fanout connections. Valid @fanout values must be integers greater than 0.
@fixed_placement	=	true, false	The @fixed_placement filter allows filtering instances based on whether placement is fixed.
@partition	=	*partitionName*	The @partition filter allows filtering instances, nets, paths, and pins based on partition name.
@placed	=	true, false	The @placed filter allows filtering instances, nets, and pins based on whether they are placed. In this context, a "placed" net means the net is routed.
@power	=, <, >	(floating point numbers > 0.0)	The @power filter allows filtering nets based on power consumption. Valid @power values must be a floating point number greater than 0.0.
@power_rank	=, <, >	(integers ≥ 0)	The @power_rank filter allows filtering nets based on level of power consumption relative to other nets. The most power-consuming net is ranked 1, the least power consuming net is ranked n. Valid @power_rank values must be integers greater than or equal to 0.
@region	=	*placementRegionName*	The @region filter allows filtering instances, nets, paths, and pins based on placement region name.
@sink_type	=	(device-dependent)	The @sink_type filter allows filtering nets and pins based on what type (cell name) of instance(s) the net is driving.
@type	=	(device-dependent)	The @type filter allows filtering instances based on type (cell name) of instance.

Table Notes

1. Some instances may be part of multiple clock domains, such as a CDC instance.

Timing Across All Temperature Corners

ACE supports place and route across all temperature corners as well as reporting timing across all temperature corners of interest for a given place-and-route result at a specific junction temperature. This feature helps the designer to confirm whether the optimized placed-and-route result is able to close timing at the desired frequency (F_{MAX}) at all temperature corners of interest.

> **Note**
>
> By default, ACE only optimizes place and route for a single user-chosen temperature corner per implementation, but then reports the timing analysis results of that routed netlist for all temperatures of interest. If it is desired to optimize place and route for multiple temperatures, the impl-option `pnr_optimize_corners` should be set to 1.

ACE requires setting the desired junction temperature at which the design is placed and routed for the target F_{MAX} (unless `pnr_optimize_corners` is not 0). This selected junction temperature is used to load the corresponding timing libraries and to optimize the place and route to close timing at the requested frequency.

If `pnr_optimize_corners` is not zero, the basic junction temperature setting is only used for timing reports up to and including the routed timing report. The final timing report always consists of multiple reports, one for each available corner.

This temperature target can be set within the Options View (see page 103) under **Design Preparation → Junction Temperature**. With the desired junction temperature selected, place and route is performed. The Final timing reports are always generated for all corners, because any given corner might be worse than any other given the underlying hardware technology used (due to the effects of temperature inversion).

> **Note**
>
> The Power Dissipation Report (see page 228)s are generated for each temperature corner as well.

When reports are generated for multiple corners, the report file names are extended with a suffix describing the PVT corner contained within the report. The suffix includes the speed grade, the voltage, and the temperature, in that order, separated by underscores: `_<speed_grade>_<voltage>_<junction_temp>`

For example:

- `_C1_1p00V_85C` corresponds to: speed grade = C1; voltage = 1.00V; junction temperature = 85ºC
- `_C2_0p70V_n40C` corresponds to: speed grade = C2; voltage = 0.70V; junction temperature = -40ºC

In addition to the above, there is an additional opportunity to apply different optimization strategies by putting the design through a Multiprocess run via the Multiprocess View (see page 83) GUI. In Multiprocess:

- Different place-and-route optimization strategies are applied
- Timing is reported across all temperature corners for each of the Multiprocess place-and-route strategy (called seed/implementation) applied
- A Multiprocess summary report is generated that lists all timing results across all temperature corners.

ECO Commands

ECO Use Cases

The ECO Command Tool Chain is a set of useful tools that allow editing a design with a high level of granularity. These tools can be used to achieve highly specific goals, such as manually adding logic blocks to the fabric, improving timing, performing analysis on the FPGA itself, and more.

Net Legality Definition

Throughout this documentation, the concept of *net legality* is mentioned frequently. For a net to be legal, all of the following must be true:

- There is exactly one and only one driving/output pin connected to the net.
- There is at least one input/sink pin connected to the net.
- All instances to which the net connects must be placed (with the exception of their respective site pins which do not need to be placed).

When a net is *legal*, the router may route the net. An *illegal* net causes the router to silently exit. When performing ECO changes, it is necessary to tie-off and/or correct any nets that become illegal. ECO commands indicate illegal nets caused by each ECO command, including why such nets were deemed illegal. It is necessary to manually keep track of such nets and correct them along the way.

Disclaimers

> **Caution!**
>
> ECO commands are intended for advanced users only. **Use at your own risk!**

ECO commands are intended to be performed at the end of the place-and-route Flow Steps (see page 221). Performing ECO before or during place-and-route is possible, but this requires more caution.

Although optional, it is highly recommended to specify the name of a site when inserting an instance, or else ACE refuses to perform net-routing with nets connected to unplaced instances. If desired, optionally execute `run_place` after instance insertion to let ACE handle placement automatically. The flow step `run_place:`

- May perform automatic placement of an ECO-inserted instance, but if the new instance is deemed redundant or useless by the ACE reconditioner, it is removed from the design.
- Is not incremental; it always places all instances in the design, which can be time consuming. As such, when using `ace_eco::insert_instance`, consider placing the new instance manually.

While performing ECO, certain nets may become partially routed (i.e., calling `rewire_net -connect` without specifying `-reroute`) or derouted (i.e., the nets lost a pin connection that kept them legal). In these circumstances, a warning is displayed on the Tcl console. All such illegal nets must be identified and resolved before proceeding.

When any illegal nets have been made legal (a driver added to a floating net, a sink pin added to a dangling net, drivers removed from a multi-driven net), `rewire_net <netName> -reroute` must be called to physically create the connections specified in the ECO-modified netlist.

If it is decided to perform `run_prepare` after performing ECO, all changes made are overwritten since `run_prepare` sources the original netlist(s) specified in the ACE project, but not any ECO modifications. As such, keep in mind that all work might be accidentally undone by calling `run_prepare`.

Running ECO commands but failing to resolve issues created by the resulting changes may potentially lead to errors within ACE if certain functions are subsequently called. For instance, a set of ECO commands could disconnect an output pin from a net, but fail to connect a new one. If another (non-ECO) command is then called that assumes that each net has a driving pin, ACE reports errors.

ECO commands modify the gate-level netlist generated from `run_prepare`, and rerunning `run_place` or `run_route` with a modified netlist may lead to unintended consequences to the design.

> **Tip**
>
> It is recommended to save often when performing ECO as inadvertently executing a wrong command may either break the design beyond repair or cause ACE to report errors, forcing a restart from scratch.

ECO Commands

The ECO commands are all in a special `ace_eco::` Tcl Namespace; they are not included in the global namespace. These commands are

- `ace_eco::delete_instance`
- `ace_eco::delete_net`
- `ace_eco::get_instance_pins`
- `ace_eco::insert_instance`
- `ace_eco::insert_net`
- `ace_eco::rewire_instance`
- `ace_eco::rewire_net`

delete_instance

```
Command Syntax

ace_eco::delete_instance {<i:instance_name> <i:instance_name> ...} [-reroute]
```

The `delete_instance` command deletes the named instances and disconnects the pins of those instances from their respective nets. Any nets left with no connections are not automatically deleted. These nets must be deleted manually using `ace_eco::delete_net`.

Table 131: *delete_instance Arguments*

Arg Name	Optional	Description
<instances>	No	List of user design instances to be deleted. The "i:" prefix is optional on each instance.
`[-reroute]`	Yes	The optional `-reroute` argument is used to re-route nets after the instances have been deleted.

delete_net

```
Command Syntax

ace_eco::delete_net {<n:net_name> <n:net_name> ...}
```

The `delete_net` command deletes the named nets.

Table 132: *delete_net Arguments*

Arg Name	Optional	Description
<nets>	No	List of user design nets to be deleted. The "n:" prefix is optional on each net.

get_instance_pins

```
Command Syntax

ace_eco::get_instance_pins {<i:instance_name> <i:instance_name> ...}
```

The `get_instance_pins` command returns a list of all pins (and nets, if connected) for the named instances. The returned lists takes the form of:

```
{{t:instance1:pin1 n:net1} {t:instance1:pin2 n:net2} {t:instance1:disconnected_pin3}} {{t:instance2:pin1
n:net1} ...} ...
```

Table 133: *get_instance_pins Arguments*

Arg Name	Optional	Description
<instances>	No	List of user design instances to be queried. The "i:" prefix is optional on each instance.

insert_instance

```
Command Syntax

ace_eco::insert_instance <i:instance_name> <cell_type_name> [-site <s:site_name>] [-pins {{<p:pin_name>
<n:net_name>} {<p:pin_name> <n:net_name>} ...}] [-parameters {{<param_name> <param_value>} {<param_name>
<param_value>} ...}] [-fixed] [-reroute]
```

The `insert_instance` command generates a new instance of the specified cell type and inserts it into the netlist. If `-site` is specified, the command places the new instance on the named site (given that the site is legal to use).

Table 134: *insert_instance Arguments*

Arg Name	Optional	Description
<instance_name>	No	The name which is given to the newly inserted instance. The "i:" prefix is optional.
<cell_type_name>	No	Type of instance. This must be a valid cell type for the current fabric.
`-site`	Yes	The optional `-site` argument names the site where the new instance is placed. The site named must be compatible with the cell type, or the command aborts before the instance is created. The "s:" prefix is optional.

Arg Name	Optional	Description
`-pins`	Yes	The optional `-pins` argument specifies a list of pin name/net name pairs. Each named pin is connected to the associated named net. The "p:" and "n:" prefixes are optional.
`-parameters`	Yes	The optional `-parameters` argument specifies a list of user parameters and values for the new instance. The parameters must be compatible with the cell type, or the command aborts before the instance is created.
`-fixed`	Yes	The optional `-fixed` argument indicates that given a -site parameter, the newly created instance is fixed to that site. By default, Soft Placement is performed (no fixing).
`-reroute`	Yes	The optional `-reroute` argument is used to re-route nets after the instance has been inserted.

Warning!

- The instance name is currently not verified to follow Verilog/VHDL identifier standards.
- Although optional, it is highly recommended to specify the name of a site when inserting an instance; otherwise, ACE refuses to route pins with an unplaced instance.

insert_net

```
Command Syntax

ace_eco::insert_net <n:net_name> {<t:instance_name:pin_name> <t:instance_name:pin_name> ...} [-route]
```

The `insert_net` command creates a new net and insert it into the netlist. This new net must have at least two connections specified, and these connections must make the net legal.

Table 135: *insert_net Arguments*

Arg Name	Optional	Description
<net_name>	No	The name to be given to the newly inserted net. The "n:" prefix is optional.
<pins>	No	List of fully-qualified (including instance) pin names. Each named pin is connected to the new net. The "t:" pin prefixes are optional.
`-route`	Yes	The optional `-reroute` argument is used to automatically route the new net after it has been inserted.

Warning!

- The new net name is currently not verified to follow Verilog/VHDL identifier standards.
- The `insert_net` command does not connect to pins already connected to a net; those pins must be disconnected first before calling this command.
- The user-specified net must be legal (has at least one input pin and exactly one driving pin) upon creation, or else the net is not created. However, instances which the new net connects to do not have to be placed.

rewire_instance

```
Command Syntax

ace_eco::rewire_instance <i:instance_name> "{<p:pin_name> [n:net_name]} {<p:pin_name> [n:net_name]} ..."
[-reroute] [-disconnect]
```

The `rewire_instance` command allows connecting or disconnecting the pins of an instance to/from specific nets. Both of these operations can be performed at the same time. Connections from user-specified nets to user-specified pins may be created, deleted, or changed.

Table 136: *rewire_instance Arguments*

Arg Name	Optional	Description
<instance_name>	No	The name of the instance whose wiring is to be changed. The "i:" prefix is optional.
<pin_net_pairs>	No	List of user design pins/ports paired with the optional nets to which each are to be connected. The "p:" and "n:" prefixes are optional. Example: `''{pin1 net1} {disconnected_pin2}''`
`-reroute`	Yes	The optional `-reroute` argument is used to re-route nets after the instance has been inserted.
`-disconnect`	Yes	The optional `-disconnect` argument causes all existing connections to be disconnected before applying new pin/net connections.

Note

- If a named pin/net connection already exists as specified, that argument is safely ignored.
- If a pin name is provided without a net name, that pin is disconnected from any currently connected net.
- If `-disconnect` is used with an empty pin_net list, all prior connections are removed without any new connections being created.

rewire_net

Command Syntax

```
ace_eco::rewire_net <n:net_name> [-connect "<p:pin_name> | <n:net_name> ... "] [-disconnect "<p:
pin_name> ..." ] [-clocktype <type>] [-reroute] [-verbose]
```

The `rewire_net` command allows connecting/disconnecting pins from the specific net. The same action could also be accomplished with `ace_eco::rewire_instance` commands, but that can quickly become very cumbersome if most of the pins on one net should now connect to another net. A single `ace_eco::rewire_net` command can do the work that would require hundreds of `ace_eco::rewire_instance` commands, each specifying the same net.

> **Note**
>
> The command `rewire_net` may be used on a net with no arguments except for `-reroute`, to perform routing on the specified net. This option is useful for cleanup work as it is necessary to route any remaining partially or unrouted nets.

Table 137: *rewire_net Arguments*

Arg Name	Optional	Description
<net_name>	No	The name of the net whose wiring is to be changed. The "n:" prefix is optional.
`-connect`	Yes	The optional `-connect` argument specifies a list of pins to be connected to a net. If already connection, they are first disconnected from their original nets.
`-disconnect`	Yes	The optional `-disconnect` argument specifies a list of pins that should be disconnected from net.
`-clocktype`	Yes	The optional `-clocktype` argument indicates the clock type of the pins to be connected.
`-reroute`	Yes	The optional `-reroute` argument is used to re-route nets after the instance has been inserted.
`-verbose`	Yes	The optional `-verbose` argument generates additional feedback as the command is running.

GUI Support

GUI support for ECO functionality is hidden by default. To enable ECO actions in the GUI, enable the checkbox found at: **Window → Preferences → User Advanced Preferences → Enable ECO Functionality**.

When enabled, ECO actions for the ECO commands appear in right-click context menus available on most views in the Floorplanner Perspective. For example, right-click a net to display the available ECO net commands; right-click an instance to display available ECO instance commands, etc.

> **Warning!**
>
> The GUI ECO wizards do not provide extra safety checks or guidance at this time. Errors, warnings, and success feedback from the ECO changes are only shown in the Tcl Console View (see page 144) as the ACE ECO Tcl commands themselves are executed by the wizards.

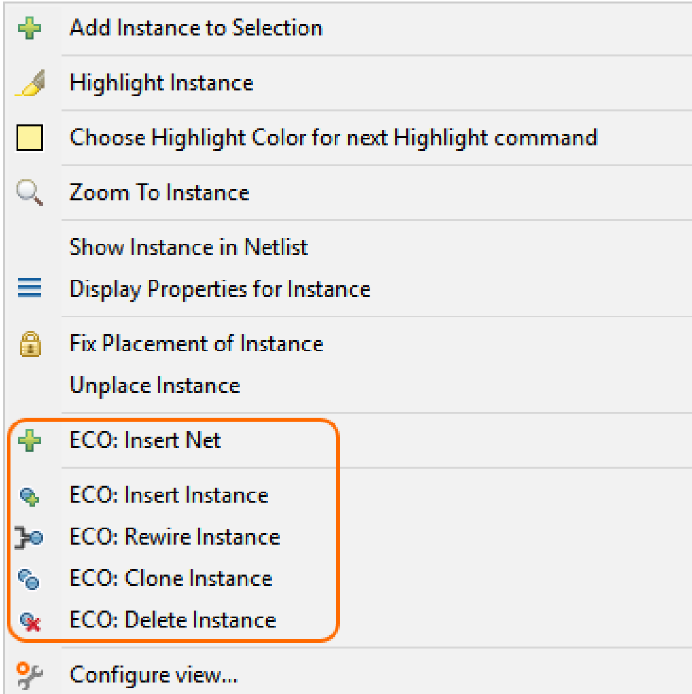


Figure 114: ***Example of ECO Instance Commands in Context Menu***

Add Instance Pin Dialog

The Add Instance Pin Dialog selects an existing pin to add to an existing instance.

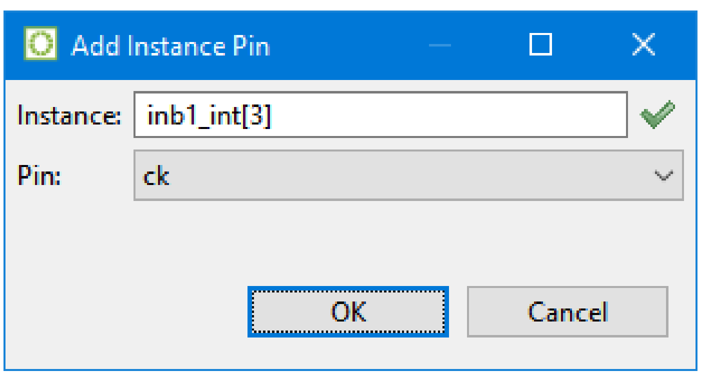


Figure 115: ***Add Instance Pin Dialog Example***

Table 138: ***Add Instance Pin Dialog Fields***

Field	Description
Instance	A valid instance name. If an invalid name is specified, the green check mark next to the Instance field changes to indicate the error.
Pin	A list of the instance pins.

ECO Insert Instance Dialog

The ECO Insert Instance Dialog allows the addition of a new instance.

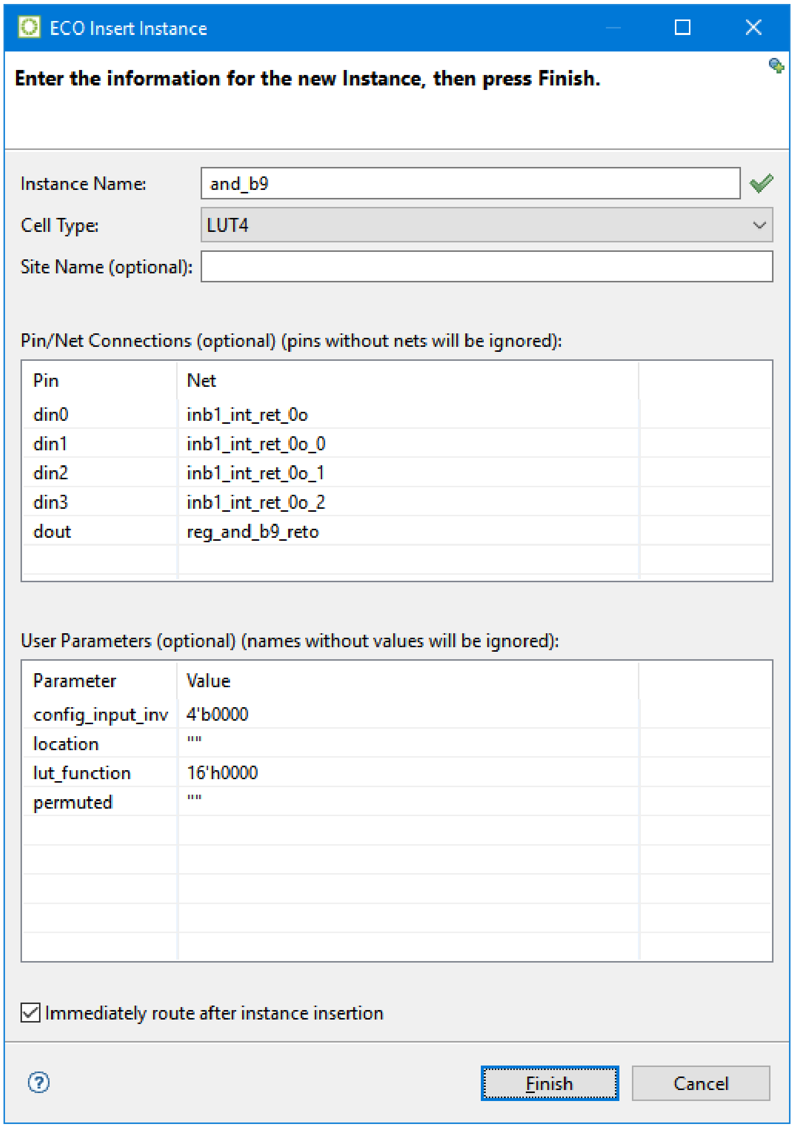


Figure 116: *ECO Insert Instance Dialog Example*

Table 139: *ECO Insert Instance Dialog Fields*

Field	Description
Instance Name	The name for the new instance. The name must be unique to the design. If an instance with the given name already exists, the green check mark next to the Instance Name field changes to indicate the error.
Cell Type	The cell type for the new instance.

Field	Description
Site Name (optional)	The name of the site where the new instance should be placed.
Pin/Net Connections	Click a cell in the Net column to connect a pin to an existing net.
User Parameters (optional)	Click a cell in the Value column to specify a parameter value. Parameters without values are ignored.
Immediately route after instance insertion	If enabled, the design is rerouted immediately after the new instance is inserted.

ECO Insert Net Dialog

The ECO Insert Net Dialog allows the addition of a new net.

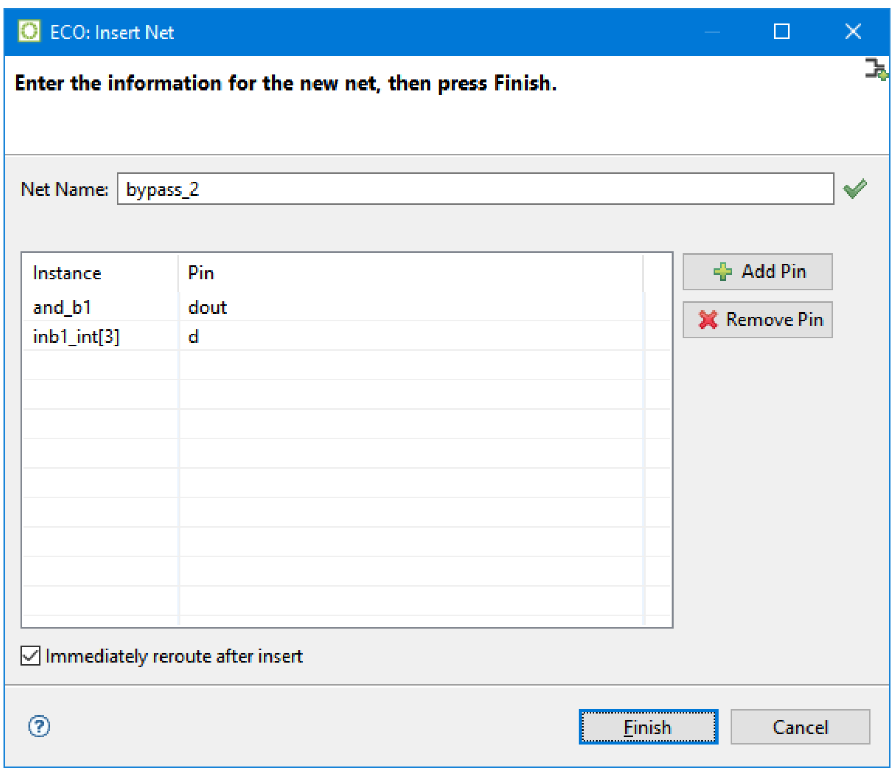


Figure 117: ***ECO Insert Net Dialog Example***

Table 140: ***ECO Insert Net Dialog Fields***

Field	Description
Net Name	The name for the new net. The name must be unique to the design. If a net with the given name already exists, the green check mark next to the Net Name field changes to indicate the error.

Field	Description
Add Pin	The Instance Pins table in the dialog lists the pins to which the new nets are connected. Use the **Add Pin** button to add pins to this table.
Remove Pin	Use the **Remove Pin** button to remove all currently selected pins from the Instance Pins table.
Immediately reroute after insert	If enabled, the design is rerouted immediately after the new net is inserted.

ECO Rewire Instance Dialog

The ECO Rewire Instance Dialog allows adjusting the properties of an existing instance.

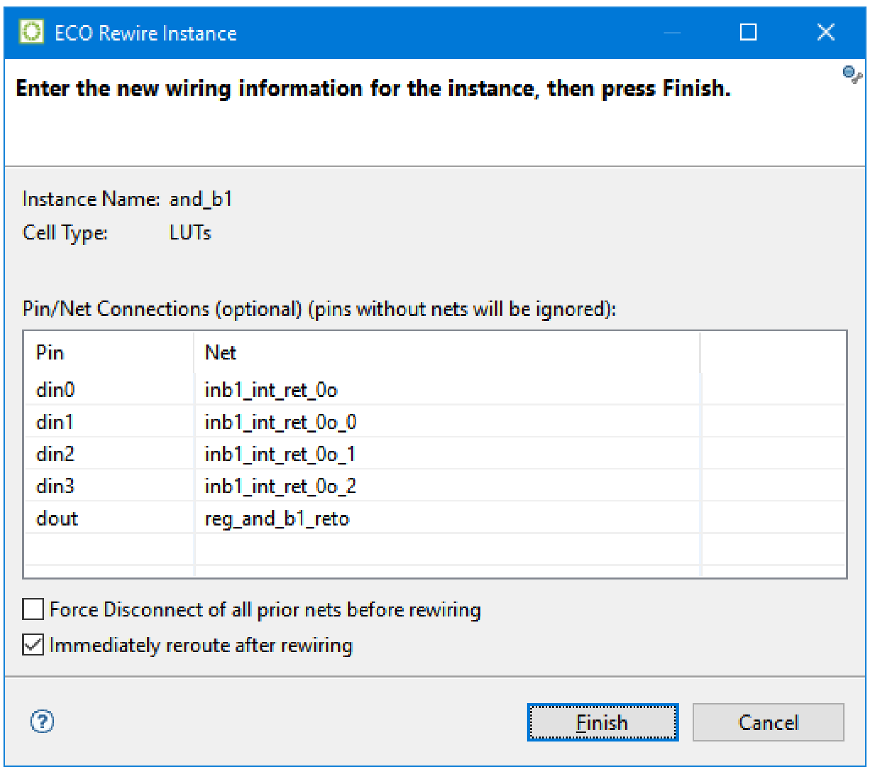


Figure 118: *ECO Rewire Instance Dialog Example*

Table 141: *ECO Rewire Instance Dialog Fields*

Field	Description
Pin/Net Connections	Click a cell in the Net column to connect the pin to a different net.
Force disconnect of all prior nets before rewiring	Causes all existing connections to be disconnected before applying new pin /net connections.

Field	Description
Immediately route after instance insertion	If enabled, the design is rerouted immediately after the new instance is inserted.

ECO Rewire Net Dialog

The ECO Rewire Net Dialog allows an existing net's pin connections to be adjusted.

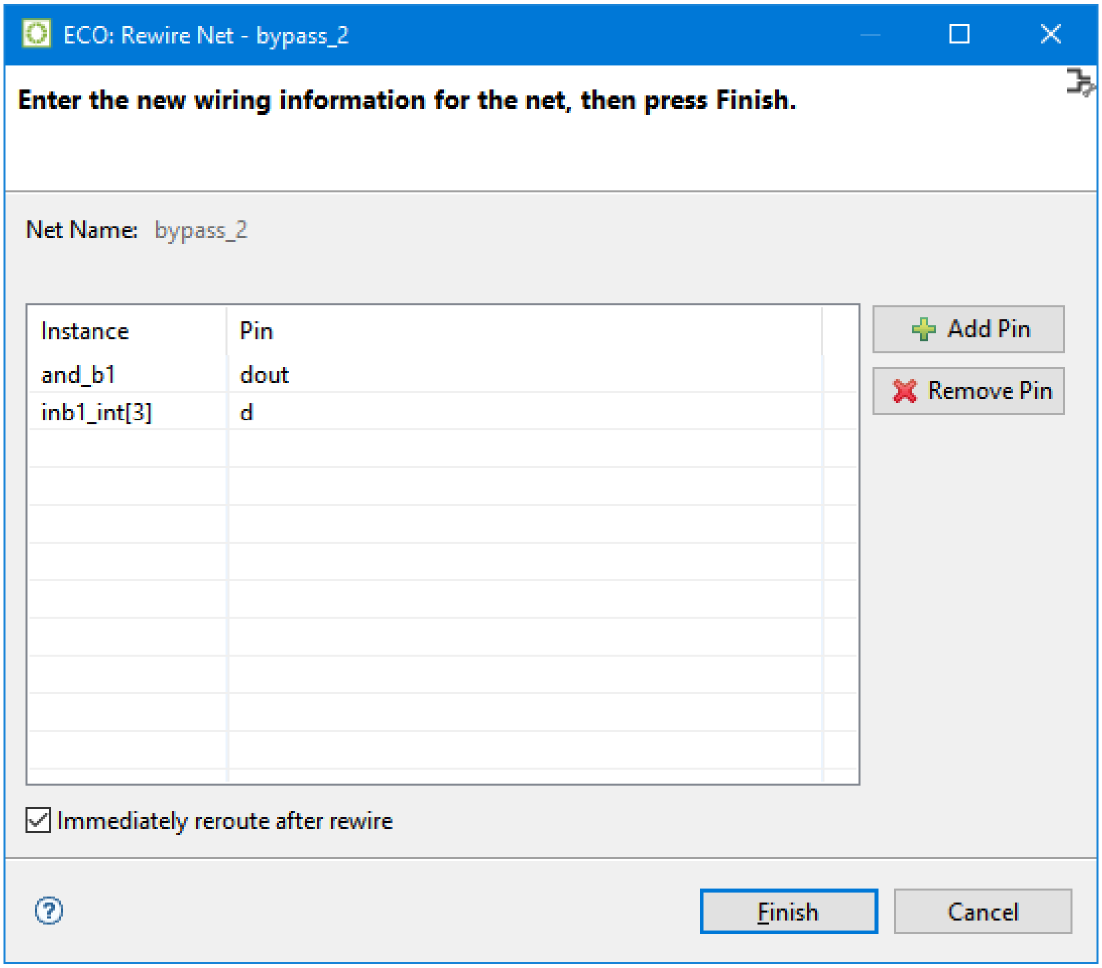


Figure 119: ***ECO Rewire Net Dialog Example***

Table 142: ***ECO Rewire Net Dialog Fields***

Field	Description
Add Pin	The Instance Pins table in the dialog lists the pins to which the new net is connected. Click this button to add pins to the table.
Remove Pin	Click this button to remove all currently selected pins from the Instance Pins table.

Field	Description
Immediately reroute after insert	If enabled, the design is rerouted immediately after the Insert Net dialog is completed.

Fabric Clusters

Device fabrics in the Achronix Speedster and Speedcore families consist of a device-specific pattern of functional resources called tiles. For example, reconfigurable logic block (RLB) tiles contain LUT, DFF, and ALU sites. BRAM tiles contain block RAM sites. The tiles are arranged in rows and columns, with all columns in the device core (not including the I/O ring) consisting of identical tile types. The arrangement of tiles into columns may look somewhat irregular (i.e., 5 columns of LUTs, a BRAM column, 5 more columns of LUTs, and an LRAM column). However, at a coarse level, those tile arrangements do follow a regular pattern. All fabrics are constructed by arranging rows and columns of tiles into a basic unit of layout called a fabric cluster. A complete device is created by replicating the fabric clusters into a larger grid of rows and columns. For example, the Speedster7t AC7t1500 has 8 rows and 10 columns of fabric clusters, for a total of 80. See the *Speedster7t FPGA Datasheet* (DS015) for additional details.

All fabric clusters are identical, containing exactly the same pattern of tile rows and columns. The exact numbers of fabric clusters, the dimensions of each cluster, and the arrangement and types of tiles within each cluster, are all specified by the chosen target device. A designer can make use of this regularity if the design consists of a number of identical cores. If the cores are designed to fit exactly within one or more fabric clusters, a complete design can be created by replicating that core multiple times across the device using the fabric cluster grid.

The use of placement regions and placement region constraints (see page 354) might be necessary to guide the placement of multi-core designs in order to align them with the fabric cluster grid. This should be discussed thoroughly with an Achronix FAE first, as improper use of placement region constraints can lower QOR or even cause placement or routing to become unsolvable.

Chapter - 3: Tasks

While the Concepts (see page 24) section was primarily concerned with which features exist in ACE, this Tasks section is concerned with how features in ACE may be best utilized.

Running ACE

ACE can be run with full functionality in three different modes:

- GUI Mode (see page 259)
- Command-line Mode (see page 259)
- Batch Mode (see page 260)

A fourth mode, Lab Mode (see page 260), is also available, with reduced functionality.

Finally, a table of typical supported ACE Startup Arguments (see page 260) is provided at the end of this section.

GUI Mode

To run in GUI mode, invoke the **ace** executable either with no options or with the **-gui** option. GUI mode launches the interactive GUI, from which all commands are issued.

Starting ACE in GUI Mode, implicit

```
% ./ace
```

or

Starting ACE in GUI Mode, explicit

```
% ./ace -gui
```

Command-line Mode

To run in command-line mode, invoke the **ace** executable with the **-b** option from a console. Command-line mode takes control of the console and allows interactively entering Tcl commands at a command prompt.

Starting ACE in Command-line Mode

```
% ./ace -b
-- ACE -- Achronix CAD Environment -- Version 5.4 -- Build  84486- -- Date 2015-02-11 19:58
-- (c) Copyright 2006-2015 Achronix Semiconductor Corp.  All rights reserved.
-- all messages logged in file /home/username/.achronix/ace_2015_02_13_11_00_11.log, created at 11:00:11
on 02/13/2015
INFO: License ace-v1.0 on server acxlicense (9 of 10 licenses available). Running on docs.achronix.local
(x86_64).
ACE>
```

Batch Mode

To run in batch mode, invoke the **ace** executable with the **-b** option and the **-script_file** option.

```
Starting ACE in Batch Mode

% ./ace -b -script_file path_to_script_file.tcl
```

Lab Mode (Reduced Functionality)

ACE also supports a reduced functionality mode, intended for use in Lab environments. The primary purpose of this mode is to allow a lighter-weight tool for chip programming and debugging, and/or for demonstrating hardware functionality with demo designs. In this mode, a license is not required (no license check occurs), and thus most Tcl functionality is unavailable (because most ACE Tcl functionality requires an appropriate license). In lab mode, the user is unable to work with project files, run the Flow, view the Floorplanner, configure IP, etc.

When the ACE GUI is in lab mode, only the default views within the Programming and Debug Perspective and HW Demo Perspective are usable. Only the subset of ACE Tcl commands needed to support those views is functional. The views within the Projects Perspective, Floorplanner Perspective, IP Configuration Perspective, and NoC Performance Perspective is non-functional (and inaccessible), since these views and editors require licensed ACE functionality.

```
Starting ACE GUI in Lab Mode

% ./ace -lab_mode
```

or

```
Starting ACE in Command-line Lab Mode

% ./ace -b -lab_mode
-- ACE -- Achronix CAD Environment -- Version 8.8 -- Build xxxx -- Date 2022-xx-xx xx:xx
-- (c) Copyright 2006-2022 Achronix Semiconductor Corp.  All rights reserved.
-- all messages logged in file /home/username/.achronix/ace_2022_08_09_10_11_12.log, created at 10:11:12
on 08/09/2022
ACE>
```

or

```
Starting ACE in Batch Lab Mode

% ./ace -b -lab_mode -script_file path_to_jtag_script_file.tcl
```

ACE Startup Arguments

The most common startup arguments are listed in the table below. Other arguments exist, but should only be used under the guidance of an Achronix FAE (or as specified in one of the Troubleshooting sections of an Achronix User Guide).

<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td><code>-b</code>
or
<code>-batch</code></td><td>Starts ACE in non-GUI mode, also known as Batch or Command-line mode.
See also: the <code>-script_file</code> argument</td></tr>
<tr><td><code>-lab_mode</code></td><td>Starts ACE in (unlicensed) Lab Mode, with limited functionality. May be used in combination with ACE GUI mode, Command-line mode, and/or Batch mode. See the prior section in this chapter for more details about ACE Lab Mode.</td></tr>
<tr><td><code>-log_file</code> <
<i>path_to_log_file</i>></td><td>Use this to override default ACE session logging behavior, which would otherwise default to creating a session log file at <<i>user_home_dir</i>><code>/.achronix/ace_</code><<i>datestamp</i>>_<<i>timestamp</i>><code>.log</code>.
A session log file covers all ACE activity across all projects and implementations touched during the process lifetime of ACE.
Keep in mind that the <code>-log_file</code> argument has no impact upon implementation log files; individual implementation log files are always created under each implementation directory whenever that implementation is being accessed. There is currently no way to override the path of the implementation log files.)</td></tr>
<tr><td><code>-print_progress</code></td><td>Enables verbose progress messages to be logged during execution of the Flow. Verbose progress logging is disabled by default.</td></tr>
<tr><td><code>-project_file</code> <
path_to_acxprj_file></td><td>Only allowed when in ACE is started in GUI mode. Allows pre-specifying which *.acxprj file should be loaded by ACE at startup. This is similar to immediately calling the following ACE Tcl command at the Tcl command prompt after ACE has started:
<pre>load_project <path_to_acxprj_file></pre></td></tr>
<tr><td><code>-script_file <</code>
path_to_script_file> [-
<code>script_args <</code>
string>]</td><td>Only allowed when ACE is started in batch mode. ACE immediately executes the script contained in the script file. If the optional <code>-script_args <string></code> is also used, then the arguments are passed to the script file when the script is executed. (1)</td></tr>
<tr><td>-
<code>snapshot_version</code>
<<i>int</i>></td><td>Allows specifying which version of Achronix Snapshot should be used during the ACE session. Version 2 was used for AC22i devices. Version 3 is the modern version, used for all subsequent devices.</td></tr>
<tr><td><code>-version</code></td><td>Intended for use in ACE Batch mode. Requests that ACE report its version number and then immediately exit. (2)
Example:
<code>Using the -version argument (in Linux)</code>
<pre>% ./ace -b -version
-- ACE -- Achronix CAD Environment -- Version 8.8 -- Build xxxx -- Date 2022-xx-
xx xx:xx
-- (c) Copyright 2006-2022 Achronix Semiconductor Corp. All rights reserved.

%</pre></td></tr>
</table>

Argument	Description

Table Notes

1. When using script files in Microsoft Windows, starting ACE in batch mode with a script file opens a new console that only survives for the life of the script file, and then immediately closes. This typically makes it difficult to read any logged results which might have appeared in that console. For best results, be sure your script logs its output to a file, or review the session log file after script execution. See also: the `-log_file` argument.
2. This argument has no effect in GUI Mode. ACE always logs the version number to the Tcl Console immediately at ACE startup in all modes. As when using the -script_file argument in Windows, ACE starts a separate console window for the session. Because ACE immediately exits after reporting/logging the version banner, this console window is only visible for a fraction of a second. The requested version information can be found in the session log file. See also: `-log_file` argument.

Working With Perspectives

Perspectives define the initial set and layout of views in the Workbench window, providing a set of functionality aimed at accomplishing a specific type of task or working with specific types of resources.

Switching Between Perspectives

Each perspective has an associated icon on the main toolbar. Switch between perspectives by clicking one of these icons.

It is also possible to view the available choices as a menu by selecting **Window** → **Open Perspective** from the menu bar.

Descriptions of the available perspectives are found in the section describing the Perspectives (see page 24) concept.

Resetting Perspectives

Often, when altering positions of Editors (see page 26) and Views (see page 31) within a perspective, this can result in an arrangement that is no longer appealing. Rather than try to manually move the Views and Editors back to the original positions, it is much faster and simpler to reset the perspective.

To restore a perspective to its original layout:

1. Select **Window** → **Reset Perspective** from the menu bar.
2. Click **OK** on the resulting dialog.

Working with Views and Editors

Views and editors are the main visual entities appearing in the Workbench. In any given perspective there is a single editor area which can contain multiple editors with a number of surrounding views providing context.

Opening Views

Perspectives offer pre-defined combinations of views and editors. To open a view not included in the current perspective, select **Window** → **Show View** from the menu bar.

Moving and Docking Views and Editors

To change the location of a view or editor in the current perspective:

1. Without releasing the left mouse button, drag the view or editor by its tab.

> **Note**
>
> A group of stacked views or editors can be dragged using the empty space to the right of the tabs.

2. While dragging the tab (or tab stack), as the mouse is moved around the Workbench, the area under the mouse changes to display (sometimes subtle) feedback indicating where the tab (or stack) can dock if the left mouse button is released at the current location.
    - Drag the tab near the left, right, top, or bottom border of another view or editor to see how that view/editor splits its available area with the dragged tab.
    - Drag the tab near the tabs of another tab stack to see where the dragged tab can be inserted/appended in the existing stack.
    - A tab may be dragged outside of the Workbench area to turn it into a detached view (a view shown in its own separate window).
3. When the view is in the desired location relative to the view or editor area under the cursor, release the left mouse button.

Table 143: *View and Editor Tab Docking Feedback*

Feedback	Description
Vertical bar between tabs	Marks the insertion point between other tabs.
Translucent rectangles overlaid upon existing view /editor	Shows the positioning of the dragged view/editor alongside the pre-existing views/editors already in that docking location.
Translucent rectangle floating outside the ACE window	Shows the position where the detached (see page 264) view/editor can appear.
	No changes. This docking location is either identical to the present layout, or is an illegal position. If the left mouse button is released, no change occurs.

> **Caution!**
>
> In Linux, there is currently a known bug, in the application frameworks underlying ACE, that may cause view /editor tab movements to detach (see page 264) instead of docking when the Help Window is open. See the Troubleshooting (see page 611) section for more details, including several workarounds.

Rearranging Tabbed Views and Editors

In addition to dragging and dropping (docking) views/editors inside the Workbench, the order of views/editors can be rearranged within a tabbed stack:

1. Click the tab of the view/editor to be moved and drag it to the desired location. As the tab is dragged across other tabs, a vertical bar insertion cursor appears.
2. Release the mouse button when the insertion cursor is in the desired location.

> **Note**
>
> A group of stacked views/editors can be moved by starting the drag using the empty space to the right of the tabs.

Detaching Views and Editors

Detached views and editors are shown in a separate window with a smaller trim. These views work like other views and editors, except that they are always shown in front of the Workbench window. To detach views/editors:

1. If the Workbench window is maximized, resize it so that it does not fill the entire screen.
2. Click and hold the tab of the view/editor to be detached.
3. Drag the tab (or tab group) outside of the Workbench window and release the mouse button. The tab can also be dragged into the window of a previously detached view/editor to have multiple detached views/editors together.

To restore the view/editor to appear inside of the Workbench window, drag its tab into the Workbench window.

Tiling Editors

The Workbench allows multiple files to be open in multiple editors. Unlike views, editors cannot be dragged outside the Workbench to create new windows. However, editor sessions can be tiled within the editor area in order to view source files side by side:

1. With two or more files open in the editor area, select one of the editor tabs.
2. While holding down the left mouse button, drag that editor tab over the left, right, top or bottom border of the editor area. The mouse pointer changes to a drop cursor, indicating where the editor session is to be moved.
3. Release the mouse button.
4. Optionally, Drag the borders of the editor area, or each editor, to resize as desired.

> **Note**
>
> This operation is a similar to moving and docking views inside the Workbench, except that all editor sessions must be contained within the editor area.

Maximizing, Minimizing, and Restoring Views and Editors

ACE provides a rich environment which, in its basic form, consists of the following:

- An Editor Area which contains one or more stacks showing the open editors
- One or more View Stacks which surround the Editor Area and contain one or more views

These elements compete for valuable screen area and correctly managing the amount of area given to each can greatly enhance your productivity within ACE.

The two most common mechanisms for managing this issue are "minimize" (use as little space as possible) and "maximize" (provide as much space as possible). ACE provides two ways to access these operations:

1. Use the minimize and maximize buttons provided on a stack border.
2. Double-click an individual tab or the blank area to the right of the tabs.

Maximize:

It is desirable at times to focus attention on one particular view or editor to the exclusion of the others. The most popular candidates for this are maximizing the editor area in order to view a report, or maximizing the Floorplanner View (see page 53) to make as much of the display available for floorplanning as possible.

ACE implements the maximize behavior by minimizing all stacks *except* the one being maximized. This allows the maximized stack to completely occupy the window while still allowing access to any open views in the perspective by using the icons located in the area around the edges of the window which are called the "Trim Stack".

Editor maximization operates on a complete Editor Area which includes all Editor Stacks, rather than simply maximizing the particular Editor Stack. This allows for "compare" workflows which require the ability to see multiple editor files in a split editor area at the same time.

Minimize:

Another way to optimize the use of the screen area is to directly minimize stacks that are of no current interest. Minimizing a stack causes it to be moved into the trim area at the edges of the workbench window, creating a Trim Stack.

> **Note**
>
> The first time a stack is minimized, the Trim Stack may end up on any edge of the window. If the Trim Stack is manually moved to a particular window edge, that same edge is typically reused when that stack is again minimized.

View Stacks get minimized into a trim representation that contains the icons for each view in the stack:

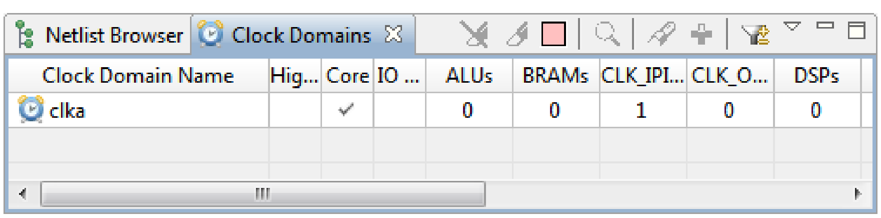


Figure 120: ***Example View Stack Before Minimization***

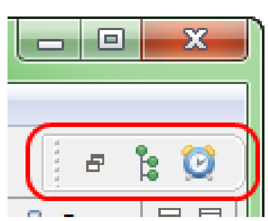


Figure 121: ***Example Trim Stack After View Stack is Minimized***

The minimize behavior for the Editor Area is somewhat different. Minimizing the Editor Area results in a trim stack containing only a placeholder icon representing the entire editor area rather than icons for each open editor (since in most cases all of the icons would be the same, making them essentially useless).

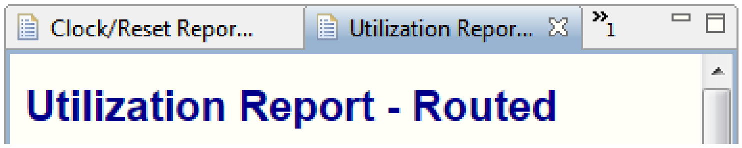


Figure 122: ***Example of Editor Area Before Minimization***

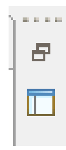


Figure 123: ***Example of Minimized Editor Area Trim Stack***

For workflows needing more than one element visible (i.e., having the Editor Area *and* a View Stack in the presentation at the same time) additional screen space can still be gained by minimizing the stacks that are not of current interest. This removes them from the main presentation and places them on the Trim Stack, allowing more space for the remaining stacks in the window.

Note

There are two ways to end up with a stack in the trim:

1. Directly by minimizing the stack.
2. Indirectly by maximizing another stack.

Depending on how the Trim Stack was created, its behavior is different: when restoring a stack from a maximized state, only those trim stacks that were automatically minimized during the initial maximize are restored to the main presentation, while stacks that were manually minimized, stay minimized.

Tip

This difference is important in that it allows fine-grained control over the presentation. While using maximize is a one-click operation, it is an "all or nothing" paradigm (i.e., no other stack is allowed to share the presentation with a maximized stack). While adequate for most tasks, it may be desired to have the presentation show more than one stack. In this case, instead of maximizing, minimize all the other stacks *except* the ones wanted in the presentation. When set up, the editor area can still be subsequently maximized, but the subsequent restore only restores the particular stack(s) that were sharing the presentation, not the ones explicitly/manually minimized.

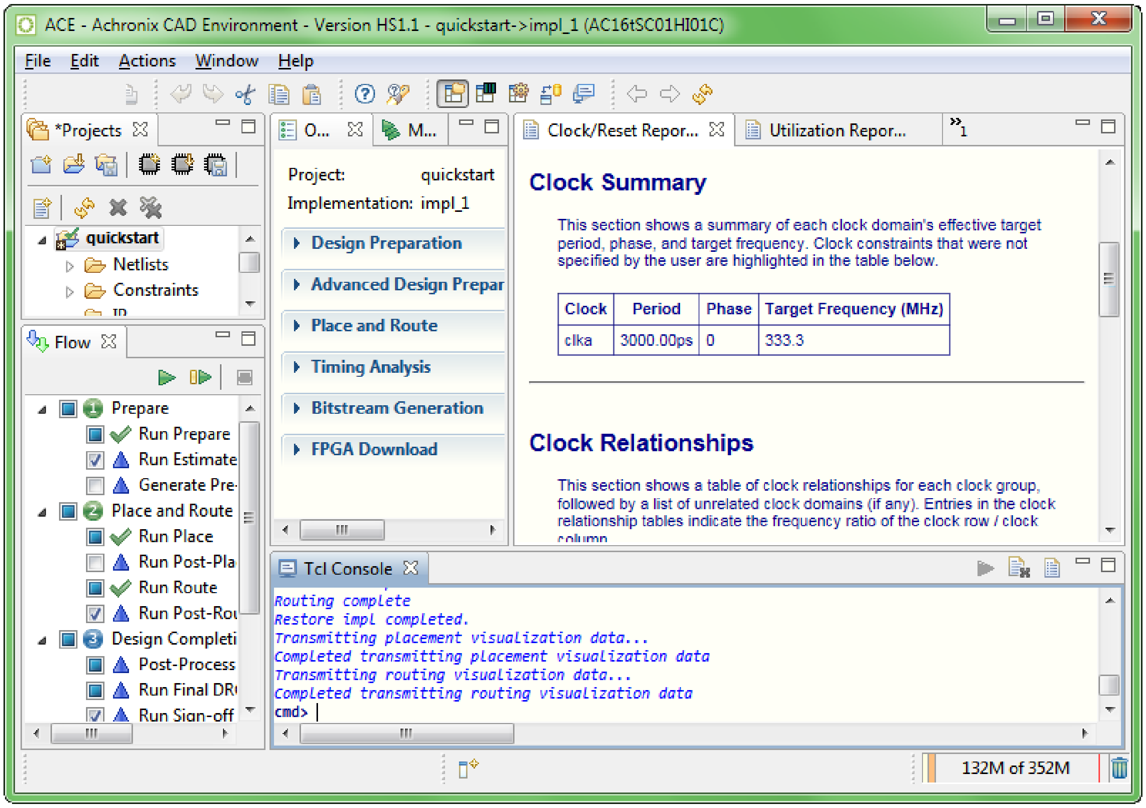


Figure 124: ***Example Default Presentation of the Projects Perspective***

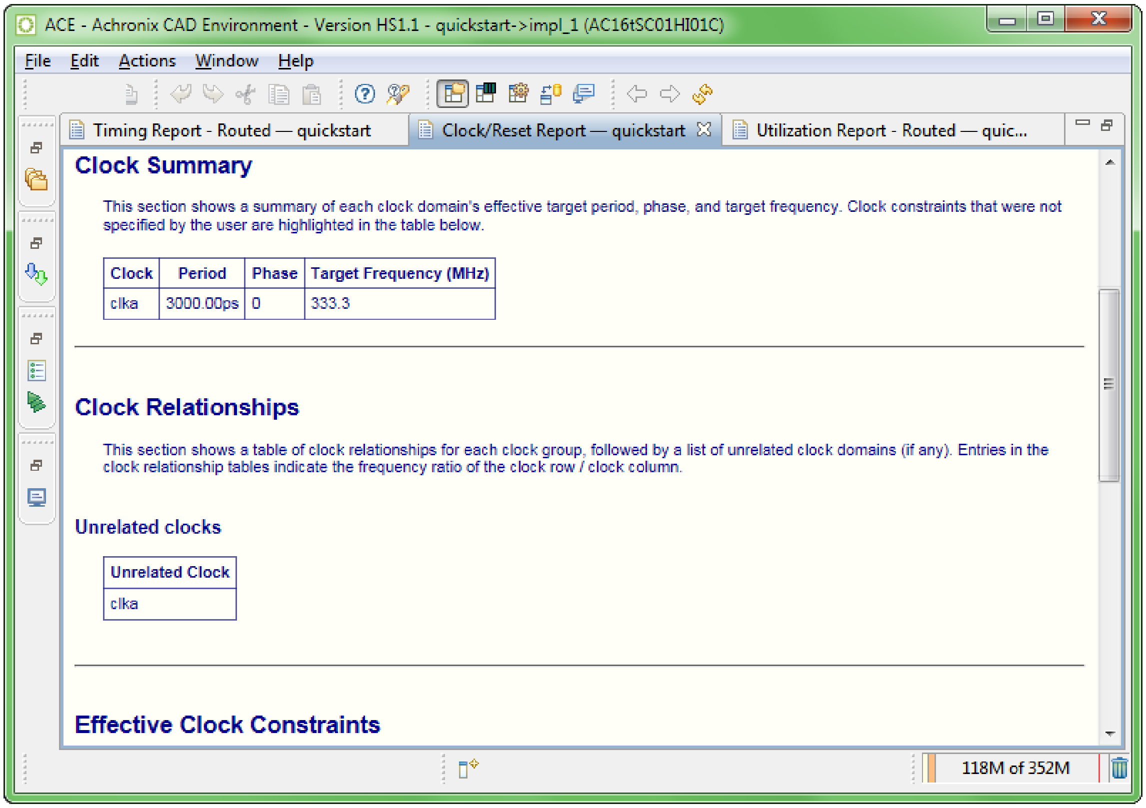


Figure 125: ***Example of the Projects Perspective With The Editor Area Maximized***

Working with Projects and Implementations

Creating Projects

To create a new project in the workspace:

1. Click the ( ) **Create Project** toolbar button in the Projects view.
2. In the Create Project dialog, type in or browse to the location of the new project directory.

> **Caution!**
>
> Directories in the path that do not exist are created.

3. Type in the new project name and click **Finish**.

After clicking **Finish**, the new project appears in the Projects view. The new project contains a default implementation named `impl_1`, which is set as the new active implementation. A project file is also created and saved in the new project directory.

Saving Projects

Some project operations cause changes to a project to be saved to the project file automatically, while others change project data without saving. Each Project with unsaved changes is marked in the GUI with an asterisk on the lower left corner of its project icon ( ). If any project in the workspace has unsaved changes, the Projects view title is also marked with an asterisk:

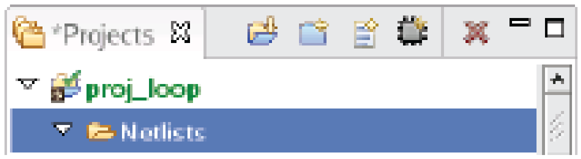


Figure 126: ***Projects View Title Unsaved Changes Example***

To save the changes to a project:

1. Select the project in the Projects view.
2. Either press **CTRL+S** on the keyboard, select the ( ) **File Save** toolbar button on the main toolbar, or select the **File** → **Save** menu option.

To save a project to a different file:

1. Select the project in the Projects view.
2. Select the **File** → **Save As...** menu option.
3. Browse to a new file location.
4. Enter a project name and click **Save**.

When exiting, ACE prompts to save changes to any projects with unsaved changes:

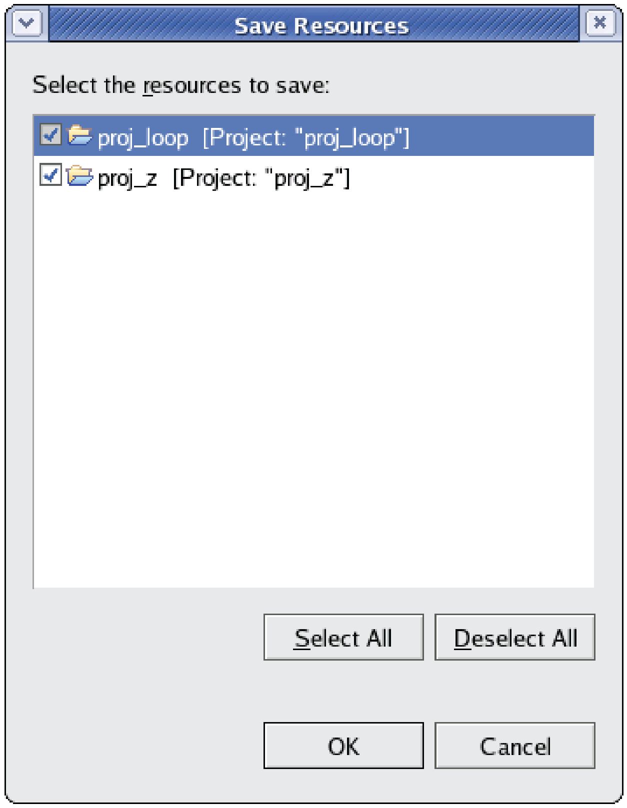


Figure 127: *Project Unsaved Changes Prompt*

Loading Projects

By default, when the ACE GUI starts, it attempts to automatically re-load all projects which were open in the prior ACE GUI session.

> **Caution!**
>
> Be aware that any projects which are still locked by another ACE session are not automatically re-loaded, nor is any related error reported. Additionally, project files from the prior session which are no longer found in the file system are not loaded, nor are any related errors reported.

Loading a Project Using the GUI

To load existing Projects (see page 215) into the workspace:

1. Click the () **Load Project** toolbar button in the Projects View (see page 125).
2. In the Load Project Dialog (see page 165), **Browse** to the location of the project directory. Or, if the project has been opened by ACE previously, find the previously opened `.acxprj` project file in the list of choices in the drop-down combo box within the dialog.
3. Select the project file and click **Open**.

After clicking **Open**, the load_project (see page 570) Tcl command is issued and the project now appears in the Projects view. This project is restored from its previous state, and its last implementation is set as the new active implementation. Any place-and-route data for the active implementation is not loaded by default. See Restoring Implementations (see page 277) for details on loading a prior place-and-route state.

> **Note**
>
> **Default Implementation Options Change Over Time**
>
> The default Implementation Options (see page 215) for ACE change over time as new optimizations become available and existing optimizations are refined. When a project is loaded from an earlier version of ACE, a "Project Version Mismatch" popup dialog is shown offering to reset all Implementation Options of all implementations to the latest default values.
>
> To avoid risking the loss of old optimizations saved in implementation options, say no to the offered reset. To see how the new default implementation options could affect the design, simply create a new implementation for the project. The new implementation contains all the new default values for implementation options, but contains no place-and-route data. Be aware that since no constraint files from the project are enabled by default in a new implementation, it is necessary to choose which constraint files to enable for the new implementation before running the flow.

Loading a Project Using Tcl

The Tcl commands load_project (see page 570) and restore_project (see page 584) may be used to open projects (and potentially also the most recent project implementation) in ACE:

- The `load_project` command is simple, and only opens the specified project for later use, without loading any additional place-and-route state of an implementation.
- The `restore_project` command is capable of much more and by default, attempts to load the most recent `.acxdb` file (potentially containing place-and-route data) for the most recent implementation in the specified project.

Project Locking and Lock Files

> **Warning!**
>
> **Project locks protect users from data corruption**
>
> ACE uses project locks and lock files to protect user data. Do not attempt to bypass ("-force") the project locks or lock files.

Achronix does not support running multiple ACE sessions on the same project (directory) simultaneously. Having a single project open in multiple ACE sessions is known to cause problems.

Every project opened by ACE is locked by that ACE session for as long as the project is open. Locking is primarily used to prevent file corruption, which could occur if multiple ACE sessions attempt to operate within the same project simultaneously. If another ACE session attempts to open a project while the project is still locked, ACE reports an error in the Tcl Console. The error message mentions the username and hostname of the session which created the project lock, allowing the coordination of sequential (not simultaneous!) project access.

Example error message for a locked project

```
cmd> load_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1"
Project: "~/output/quickstart/quickstart.acxprj" is locked by another ACE session and cannot be loaded.
This project is locked by user: TestUser1 on host: TestStation1. [...]
```

Instead of forcing project lock overrides or deleting lock files, when needing simultaneous access to a design, consult your Achronix FAE. Some potential options include Using Incremental Compilation (Partitions) (see page 363), or using version control tools to store the project.

> **Caution!**
>
> While it is possible to keep multiple copies of the same project in separate project directories, this method is extremely difficult to coordinate, and is thus not recommended by Achronix.

In the unlikely occurrence of an ACE crash, a project may mistakenly remain locked after ACE has closed. Because the project is still locked, subsequent attempts to load the project fail with an error message similar to the one above. To recover from such situations, see "Unable to Load Project: Project is Locked (see page 613)" under Troubleshooting (see page 611).

Removing Projects

To remove a project from the workspace:

1. Select a project in the Projects view.
2. Click the (✖) **Remove** toolbar button in the Projects view.

After clicking **Remove**, the project no longer appears in the Projects view. The project files are not deleted from the file system during this operation, and it is the responsibility of the user to clean up unwanted files on the disk. The project is left untouched on the disk so that it can be loaded again, later, if desired.

Opening Project Files in an Editor

To open a project file in the editor area, double-click the project in the Projects view. The project file now appears in a text editor in the editor area. Editing a project file in the workspace does not affect the project unless the project is removed and then re-loaded from the changed project file.

Adding Source Files

To add source netlist and constraint files to a project in the workspace:

1. Select the Project (see page 215) in the Projects View (see page 125) to which the source files are to be added.
2. Click the ( ) **Add Source Files** toolbar button in the Projects view.
3. In the Add Source Files Dialog (see page 147), browse to the location of the source files.
4. Select the desired file in the dialog, and click **Open**.

> **Caution!**
>
> By default, ACE loads source files in the same order they were added to the project. If ACE is loading files in an incorrect order, drag and drop them into the desired order within the project Netlists and/or Constraints nodes in the Projects View (see page 125).

After clicking **Open**, the source files appear in the appropriate netlist or constraints folder under the selected project in the Projects view. The source files are not actually loaded into the design until the **Run Prepare** flow step is run (or `run_prepare` is called). Adding a source file to a project simply creates a link to the file so that it may be loaded during flow execution.

See also:

- add_project_netlist (see page 535)
- add_project_constraints (see page 534)

- add_project_ip (see page 535)
- Removing Source Files (see page 275)
- enable_project_constraints (see page 550)
- disable_project_constraints (see page 545)

Source File Load Order

> **Note**
>
> Source file load order is shared by all Implementations (see page 215) within a given Project (see page 215). Enabling of constraint files (choosing which constraint files are actually loaded) is allowed to differ in each implementation and is managed by the checkboxes in the Options View (see page 103).

To assist with the understanding of the load order of the source files, the netlist and constraint files are listed in the Projects View (see page 125) in the same order in which they are loaded (the constraint files are additionally listed in order within the Options View (see page 103)).

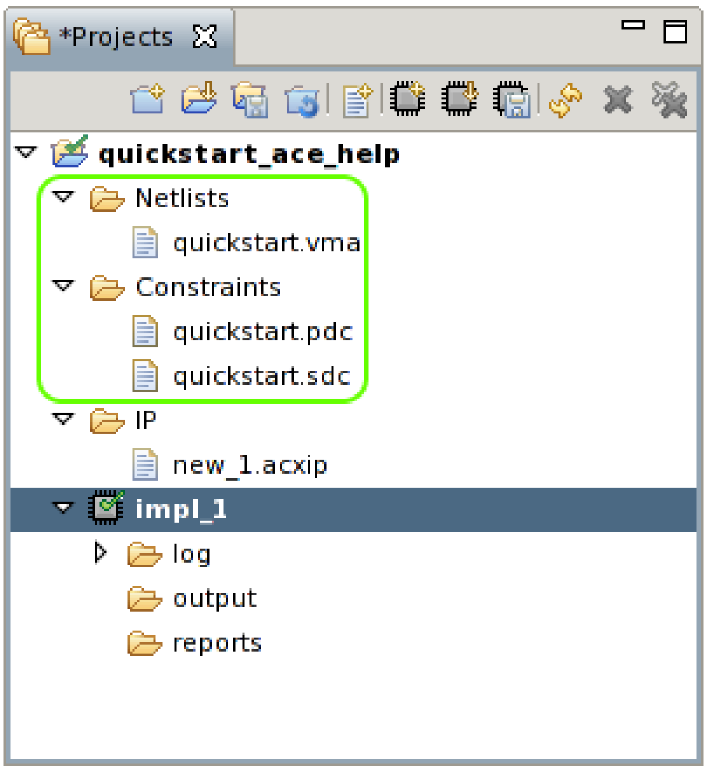


Figure 128: ***Projects View File Order Example***

By default, ACE loads source files in the same order they were added to the Project. Frequently, the order in which source files are loaded is important. For example, the creation of a clock may occur in the source file `create_clocks.sdc`, while operations upon that created clock may happen in the source file `alternate_clocks.sdc`. To avoid errors, first add `create_clocks.sdc` to the project as a source file, then add `alternate_clocks.sdc` as a source file. If attempting to add all the files to the project in a single operation, the results are platform dependent, but often the operating system "helpfully" sorts the bulk-added files alphabetically behind the scenes, which causes ACE to add them to the project in a potentially incorrect order (and thus later try loading them in that same incorrect order).

When the displayed source file load order is incorrect, there are a few ways to alter the load order:

- Changing the order of existing netlist source files and constraint source files can be performed quickly using mouse drag-and-drop operations in the Projects View (see page 125) (or by using Tcl commands created explicitly for this purpose). The tree should be expanded to show all the netlist and/or constraint files, then drag-and-drop the files to re-order them within the appropriate Project View node until they achieve the desired order. The next time the Flow is Run (see page 280), the constraint files are loaded in the chosen order.
  See also:
  - get_project_netlist_files (see page 565)
  - move_project_netlists (see page 571)
  - get_project_constraint_files (see page 564)
  - move_project_constraints (see page 571)
- A more tedious way to alter the order (but possibly the easiest way to script) is by removing all the constraint source files from the project.
  see:
  - Removing Source Files (see page 275)
  - remove_project_netlist (see page 576)
  - remove_project_constraints (see page 575)
  - remove_project_ip (see page 575)

  Add them to the project again, one at a time, in the desired order.
  See:
  - add_project_netlist (see page 535)
  - add_project_constraints (see page 534)
  - add_project_ip (see page 535))

Enabling/Disabling Constraint Files for Implementations

Implementations (see page 215) are allowed to individually enable and disable the loading of constraint files within their owning Project (see page 215). This selective loading is managed through the Options View (see page 103), under the **Design Preparation** category of Implementation Options. Simply uncheck the checkbox next to the constraint files which should not be loaded for the implementation.

See also:

- disable_project_constraints (see page 545)
- enable_project_constraints (see page 550)

Constraint files which are disabled (unchecked) for the current Active Project and Implementation (see page 221) are displayed in grey (instead of black) within the Projects View (see page 125).

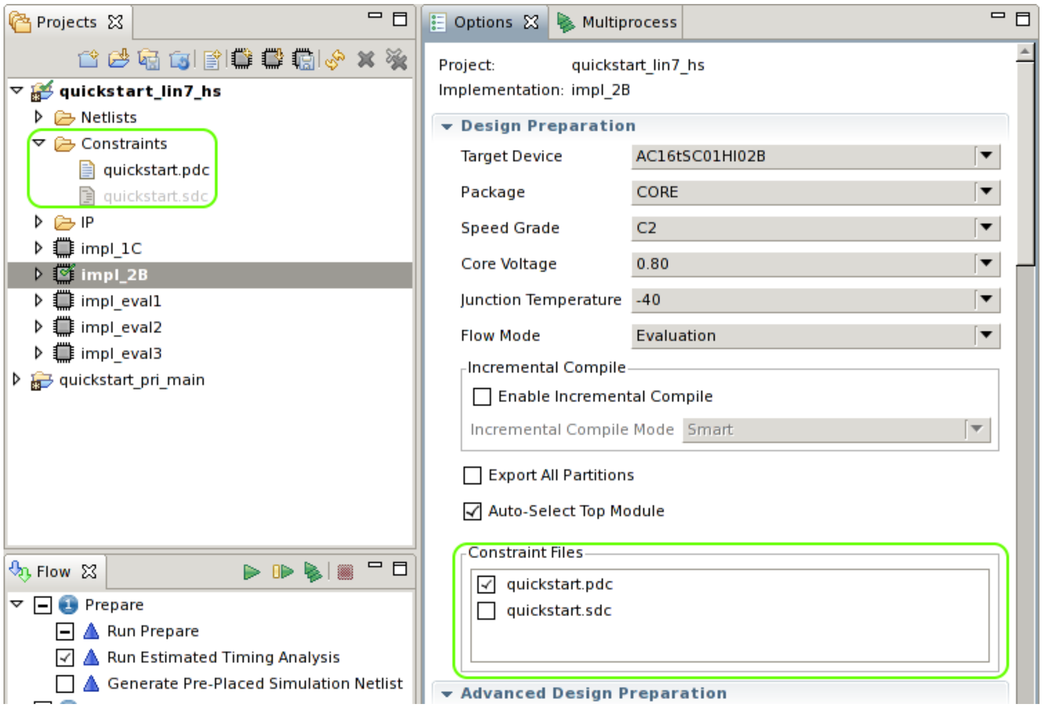


Figure 129: *Projects View Disabled Constraints File Example*

Removing Source Files

To remove a source file from a Project (see page 215) in the workspace:

1. Select a source file in the Projects View (see page 125).
2. Click the ( ) **Remove** toolbar button in the Projects view.

Or:

1. Right-click the source file in the Projects View.
2. Choose ( ) **Remove** in the popup context menu.

After clicking **Remove**, the source file no longer appears in the Projects view. Source files are not deleted from the file system during this operation, and it is the responsibility of the user to clean up unwanted files on the disk. The source file is left on the disk so that it can be loaded again later, if desired.

See also:

- remove_project_netlist (see page 576)
- remove_project_constraints (see page 575)
- remove_project_ip (see page 575)
- Adding Source Files (see page 272)

Disabling Constraint Files

It is often not necessary to completely remove constraint files from a project. Instead, constraint files can be individually disabled for any Implementations (see page 215) within a project.

1. Select/activate an implementation within the Projects View. The Options View (see page 103) is updated to show the implementation options for that implementation. At the bottom of the **Design Preparation** implementation options category, there is a list displayed of the Constraint Files for the project.
2. In the Options View, expand the **Design Preparation** implementation options category. At the bottom of the category, there is a list displayed of the Constraint Files in the project.
3. Deselect (clear) the checkbox(es) of constraint files which should not be loaded for the implementation.

See also:

- disable_project_constraints (see page 545)
- enable_project_constraints (see page 550)

Opening Source Files in an Editor

To open a source file in the editor area, double-click the source file in the Projects view. The source file now appears in a text editor in the editor area. Editing a source file in the workspace does not affect the results of the flow unless the flow is re-run on the affected project implementations.

> **Note**
>
> IP (`.acxip`) files that are not part of the currently active project cannot be opened.

Creating Implementations

To create a new implementation (see page 215) in a project (see page 215) in the workspace:

1. Select a project in the Projects view (see page 125).
2. Click the ( ) **Create Implementation** toolbar button in the Projects view.
3. In the Create Implementation dialog (see page 158), type in the name of the new implementation and click **Finish**.

After clicking **Finish**, the new implementation appears under the selected project in the Projects view. The new implementation is set to be the active implementation (see page 221) and contains default values for all implementation options (see page 215). A new implementation directory structure is also created under the project directory if it does not already exist.

Saving Implementations

To save the state of the database (options, netlist, constraints, placement, and routing data) for an implementation in a project in the workspace:

1. Activate an implementation in the Projects View.
2. Run the flow (at least through Run Prepare).
3. Optionally edit placement or routing information.
4. Click the ( ) **Save Implementation** toolbar button in the Projects view.
5. In the Save Implementation Dialog (see page 169), type the file path to the `.acxdb` Archive File to which the implementation data is to be saved and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation is stored in the `.acxdb` Archive file, which can be restored again later.

See also:

- Restoring Implementations (see page 277)
- save_impl
- restore_impl

> **Note**
>
> **Some Flow Steps Automatically Save the Implementation State**
> A subset of the Flow Steps (see page 221) automatically saves the current state in .acxdb files. These files are called:
>
> - <implementation_name>_prepared.acxdb
> - <implementation_name>_placed.acxdb
> - <implementation_name>_routed.acxdb
>
> These files are created at the end of the Run Prepare, Run Place, and Run Route flow steps, respectively.

Restoring Implementations

To restore the state of the database (options, netlist, constraints, placement, and routing data) for an implementation in a project in the workspace:

1. Activate an implementation in the Projects View.
2. Click the ( ) **Restore Implementation** toolbar button in the Projects view.
3. In the Restore Implementation Dialog (see page 167), enter the file path to the .acxdb Archive File from which to restore the implementation data and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation is restored from the .acxdb Archive file.

> The Run Prepare, Run Place, and Run Route flow steps automatically save checkpoint .acxdb files (by default) that may be restored later.

See also:

- restore_impl (see page 583)
- save_impl (see page 597)

Copying Implementations

To create a new implementation (see page 215) that is a copy of an existing implementation,

1. In the Projects View (see page 125), select (activate (see page 221)) the implementation to be copied.
2. Select the ( ) **Create Implementation** toolbar button in the Projects View.
3. In the pop-up Create Implementation dialog (see page 158):
    a. Enter the name of the new implementation.
    b. Check the **Copy Option Values from Active Implementation** checkbox.
    c. Click **Finish**.

After clicking **Finish**, the new implementation appears under the selected project in the Projects view. The new implementation is set to be the active implementation (see page 221) and contains implementation options (see page 215) values copied from the source implementation. A new implementation directory structure is also created under the project directory if it does not already exist.

Setting the Active Implementation

To change the active implementation (see page 221) in the GUI, do one of the following:

- Click an implementation in the Projects view, which activates the selected implementation
- Click a project in the Projects view, which activates the first implementation of the selected project

Changing the active implementation causes the flow status to be cleared and changes the target for all flow operations to the new active implementation.

Removing Implementations

To remove an implementation from a project in the workspace:

1. Select an implementation in the Projects view.
2. Click the ( ) **Delete** toolbar button in the Projects view.

After clicking **Delete**, the implementation no longer appears in the Projects view. Removing an implementation from a project causes all settings for the implementation to be deleted from the project file when the project is saved.

Configuring Implementation Options

To configure implementation (see page 215) options in the workspace:

1. Select an implementation in the Projects view (see page 125), changing the active implementation to the selection.
2. In the Options view (see page 103), use the controls to configure the available implementation options for the active implementation.

After changing implementation options in the Options view (see page 103), the flow status (see page 225) is cleared. A change to an implementation option requires the flow (see page 221) to be re-run for that implementation in order for the changes to affect the results of the flow. The changes to the implementation options are not saved until the affected project is saved.

Opening Output Files in an Editor

To open an output file in the editor area, double-click the output file in the Projects view. The output file appears in a text editor in the editor area.

> **Note**
>
> Editing an output file is *NOT* recommended.

Opening Report Files in an Editor

To open a report file in the editor area, double-click the report file in the Projects view. The report file now appears in a web browser in the editor area.

Note

Editing a report file is *NOT* recommended.

Cleaning Projects

In the Projects (see page 215) Perspective, while running the Flow (see page 221) for active Implementations (see page 215), ACE generates output files and sub-directories under the active implementation directory. When the configuration is changed and the entire flow or a sub-flow is re-run, any previously generated output files with the same filenames are overwritten. However, some organizations prefer to clean the active implementation directory (and sub-directories) before every flow run to avoid having any lingering stale files.

ACE provides a simple and easy way to delete (clean) these sub-directories and output files through the `clean_project` Tcl command and/or related actions in the Projects View (see page 125). Specifically, the implementation sub-directories which are cleaned include:

- `.debug/`
- `output/`
- `reports/`

Additionally, the `*.acxdb` files for the selected implementation(s) are deleted.

Warning

Cleaning projects (or implementations) is an irreversible action – the files are deleted from the file system.

In contrast, see also Removing Implementations (see page 278), which removes an implementation from a project without deleting any files from the file system.

Note

Multiprocess Reports are a special case

The Multiprocess Summary Report (see page 238) files are a special case. These files are not stored at the Implementation level, but at the Project level. Thus, these reports are not deleted when individual Implementations are deleted (**Clean Implementation**), but are deleted when an entire Project is cleaned (**Clean Project**).

Note

Log files are never cleaned/deleted

To ensure the full history of an implementation is always maintained, cleaning a project or implementation never deletes `*.log` files (or any other files found within the `<implementation_directory>/log/` subdirectories).

To clean the implementations of a project from the workspace:

1. Select a project in the Projects view.
2. Right click the selected project and select ( ) **Clean Project** from the menu.
3. This option selects all implementations of the selected project by default. Choose one or more implementations to clean from the **Clean project** dialog.

4. Click **OK**.

This operation can also be performed on an implementation:

1. Select an implementation from a project in the Projects view.
2. Right click the selected implementation and select ( ) **Clean Implementation** from the menu.
3. This option selects only the current implementation by default while the rest of the implementations for the selected project are not selected. Choose one or more implementations to clean from the **Clean project** dialog.
4. Click **OK**.

See also:

- Projects View (see page 125)
- `clean_project`
- `remove_project`
- `remove_impl`
- Removing Projects (see page 272)
- Removing Implementations (see page 278)

Running the Flow

A flow can only be run on the current Active Implementation (see page 221). If no active implementation is set in the Projects View (see page 125), then the Flow Steps (see page 221) in the Flow View (see page 61) are disabled. Some flow steps are optional while others are required. Optional flow steps may be enabled or disabled in the Flow view by checking or clearing the checkbox to the left of each flow step label.

Running the Entire Flow

To run the current Active Project and Implementation (see page 221) through the entire flow (sequentially run each of the Flow Steps (see page 221) in order):

1. Enable the desired optional flow steps (and disable the unwanted optional flow steps) by clicking the checkboxes next to the flow steps. Required flow steps cannot be disabled.
2. Choose the ( ) **Run Flow** or **Re-Run Flow** action in the Flow View (see page 61), either as a toolbar button or context menu choice.
3. (Optionally) Stop the flow from continuing to the next flow step at any time by clicking the ( ) **Stop Flow** toolbar button in the Flow view. When this occurs, ACE displays a dialog allowing the optional restoration of a prior flow state for the implementation by loading an `.acxdb` file.
   See also:
   - `restore_impl`
   - Load Acxdb Dialog (see page 182)

Disabled flow steps are skipped (not executed) during this operation.

As each individual flow step is run, its Flow Status (see page 225) changes from incomplete, to running, to either error or complete. If an error occurs during the execution of a flow step, the flow is stopped, and no further steps are attempted.

See also:

- `run`

- `enable_flow_step`
- `disable_flow_step`

> **Note**
>
> **Special note regarding the Flow and Incremental Compilation:**
> When Incremental Compilation is enabled, it might be necessary sometimes to recompile all partitions. This operation can be performed by managing the individual partitions using the Partitions View (see page 117), but an easier way to trigger the recompile is to select the Flow View context menu choice **Re-Run Flow with "-ic init"**, which re-initializes the state of all partitions before starting the full flow.
> See Using Incremental Compilation (Partitions) (see page 363) for more details. See also: `run -ic init`.

> **Note**
>
> **Special note regarding Evaluation Flow Mode:**
> When the **Flow Mode** implementation option is set to **Evaluation**, the flow steps under **Design Completion** and **FPGA Programming** are not executed. See Flow Mode (see page 226) for more details.

Running a Sub-Flow

When using the Flow View (see page 61), there are several ways to run a subset of the available Flow Steps (see page 221) on the Active Project and Implementation (see page 221).

It is possible to run individual flow steps one-at-a-time, to run all required flow steps up to a specified step (stopping when the specified step is completed), and to resume running a partial flow to flow completion.

As each flow step is run, its Flow Status (see page 225) (as displayed in the Flow View) visibly changes from incomplete, to running, to either complete or error. If an error occurs during the execution of a flow step, the flow stops running any further steps. Disabled Flow steps are not executed during these operations.

Run an Individual Flow Step

Simply right-click the chosen flow step, and select the **Run Selected Flow Step** context menu item. Alternately, double-click the chosen flow step.

If any prerequisite required flow steps have not yet been executed, they are run in standard order prior to the chosen step. Any preceding optional steps are not run, even if they are enabled.

After any prerequisite required steps are complete, the chosen flow step is executed.

> **Warning!**
>
> **Run Selected Flow Step** runs the selected step even if that step is optional and not currently enabled (its checkbox is unchecked). Also, this action executes not only the selected step, but any preceding required steps.
> Again, only the preceding *required* flow steps are run, not any preceding optional steps, even if they were selected (had their checkboxes checked).

See also:

- `run -step <id>`

Run Remaining Enabled Flow Steps (Resume Flow)

When the flow has been stopped before completion, or when a partial flow state has been loaded (see page 270) from a saved `.acxdb` file, ACE can continue the flow if the ( ) **Resume Flow** action is chosen. This action causes ACE to start running at the first enabled flow step which follows the latest successfully completed flow step.

1. Ensure the desired optional steps are enabled (checked) in the Flow View.
2. Choose the ( ) **Resume Flow** action from the view toolbar, or from the right-click context menu.

> **Note**
>
> If the current flow mode is set to **Evaluation**, the flow stops after the **Place and Route** category completes. See Flow Mode (see page 226) for details.

See also:

- `run -resume`
- `enable_flow_step`
- `disable_flow_step`

Stopping the Flow

At any time while a flow step is running, it is possible to ask ACE to stop running the flow with the Flow View ( ) **Stop Flow** action.

Some flow steps might respond by stopping immediately, while others need to perform some additional work before exiting the flow step. In both cases, the Flow Status (see page 225) of that step typically is changed to the ( ) Error status to indicate that the flow step did not complete successfully.

It is frequently the case that when the flow is interrupted in this manner, the Tcl Console shows many logged error messages for the interrupted flow step. Typically the ( ) **Resume Flow** or **Run Selected Flow Step** can be selected and ACE resumes normal work from the last successfully completed flow step.

Running Multiple Flows in Parallel

Normally, ACE only allows a single project (see page 215) implementation (see page 215) to be run through the flow (see page 221) at a time. Using the Multiprocess View (see page 83), ACE allows running multiple implementations *within a single project* through the flow in parallel, via a configurable number of parallel processes. Executing multiple implementations in this manner allows ACE to provide a Multiprocess Summary Report (see page 238) of the resulting frequencies, permitting QOR performance comparisons between implementations utilizing different starting clock constraints, placement constraints, and potential optimizations.

Finding the Multiprocess View

To make use of the (hidden by default) Multiprocess View, the view must first be made visible. To show the Multiprocess View, select the ( ) Projects Perspective (see page 24). Then, in the Flow View (see page 61), select the ( ) **Show Multiprocess View** button. This causes the Multiprocess view to be displayed, and also hides/minimizes the ACE Editor Area (where reports are displayed) to allow sufficient screen area for the Multiprocess view. The next time an ACE report is generated/opened, the ACE Editor Area again becomes visible.

Alternately, the Multiprocess view may be displayed without side-effects from within any perspective by selecting **Window → Show View → Other... → Achronix → Multiprocess**.

Configuring the Execution Queues

Within the Multiprocess view, the Execution Queue Management (see page 85)" section allows the configuring the desired number of parallel processes used to consume the queue of selected implementations. Simply set the value of **Parallel Job Count** to the desired number of parallel processes. Using the minimum value of **1** causes all queued implementations to be executed sequentially, one after another.

ACE may be configured to execute the parallel processes in the background on the host workstation running the ACE GUI, or ACE may submit each implementation as an independent executable job to an external cloud/grid/batch job submission system. Detailed configuration of the external job submission command is handled on the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201).

License Management Considerations with Multiprocess

> **Warning**
>
> Each parallel ACE process needs access to an ACE software license.
>
> **Floating licenses:**
> When running using the Multiprocess View in the ACE GUI, to run *N* parallel execution queues, *N*+1 ACE licenses are needed (the extra license is for the GUI itself, as it is managing all the queues running in the background). Talk to your Achronix FAE to ensure that your site has enough licenses to enable running with Multiprocess functionality.
>
> **Node-locked licenses:**
> When running using the Multiprocess View in the ACE GUI, provide a node-locked license for every host machine running ACE. When running local/background execution, a single node-locked license is sufficient for all ACE sessions running on that host. When running using an external job submission system, every execution host needs its own node-locked license installed on that execution host. Talk to your Achronix FAE to ensure that your site has enough licenses to enable running with Multiprocess functionality.

The following is a common best practice when determining the needed ACE floating license counts to support multiprocess runs at a specific site, as well as choosing the best value for **Parallel Job Count** based upon the available floating license count.

- Start with the number of ACE users (*U*).
- Determine the maximum number (*P*) of parallel job execution hosts available to the job submission system. Alternately, if job execution hosts are each allowed to run more than one job at a time, determine the maximum number (*P*) of ACE multiprocess jobs the system could theoretically handle in parallel, which is usually determined by ACE memory requirements.

  > **Note**
  >
  > Remain aware that ACE memory requirements vary widely based upon design size/complexity, target device, and other factors. Remember that ACE logs its peak memory consumption at the completion of every flow step — this peak memory value is a useful guideline when determining expected multiprocess memory consumption.

Sites trying to minimize license usage, or where users must share the available execution hosts equally, the minimum number of required ACE licenses (L_{min}) would then be $L_{min} = U + P$. Each user is then allowed to consume up to L_{user} licenses during their multiprocess sessions, where $L_{user} = 1 + (P \div U)$.

Sites that want to maximize job throughput, where individual users may be allowed to completely saturate the execution hosts, the maximum number of required ACE licenses (L_{max}) would then be $L_{max} = U + (P \times U)$. Each user is then allowed to consume up to L_{user} licenses during their multiprocess sessions, where $L_{user} = 1 + (P \times U)$.

Each user must then set their **Parallel Job Count** to their personal value of $L_{user} - 1$ (one license is reserved for the ACE session coordinating Multiprocess), which should then ensure that no multiprocess jobs run out of licenses.

Important Considerations When Using Background Execution on the Local Host Workstation

Be aware that if the configured number of parallel processes is too high, total execution time actually takes longer than it would at lower values. The constraints are available memory and available processor cores, as well as the load from other processes running on the host workstation.

When choosing how many parallel background implementations to allow, it is very important that users ensure they do not exhaust the physical memory (RAM) available on the executing workstation, otherwise flow execution times quickly increase (due to the OS swapping memory pages to disk). Do take into account any other users on the same workstation, as well as the memory currently in use by the already-running ACE GUI and associated back-end `acx` process.

Each additional background ACE process takes multiple Gigabytes (GB) of memory — the exact amount varies depending upon the size of the design and the size of the target Achronix device (smaller designs and smaller devices, of course, take less memory). An estimate for large designs on a very large FPGA device is around 16GB of memory used for each background process. Again, this is only an estimate — designs nearing 100% device utilization may require more memory.

Be aware that with modern multi-core hyper-threading workstations, memory limits are usually the reason to constrain the parallel process count. It is not unusual to find workstations capable of running 8 simultaneous threads while only having 32GB of RAM. While on this example workstation, if the ACE user is running the flow on a very large FPGA design (where our estimate was around 16GB per background process), the most efficient parallel process count would likely be **1** or **2**; it would depend upon the Operating System, how much memory ACE and other currently-running processes are already using, and whether the user planned to continue using the workstation interactively while the background processes were executing. Since multiple iterations through the flow are likely, it may be worthwhile to track the total multiprocess duration at multiple parallel process counts, so as the user continues working, they can use the most efficient settings for that workstation.

In the majority of cases, the parallel process count should *at most* be the *lesser* of the following two values (remaining aware that lower values may be even faster):

- **processor constraint: 1 +** T
  where
  T = the total number of simultaneous threads supported by the workstation,
  $T = (P \times (C \times H))$, where
  P = the total number of processors in the workstation
  C = the number of physical cores per processor
  H = 2 if the cores are hyper-threaded, 1 if not

- **memory constraint: *A* / *D***
  where
  D = amount of memory needed by the design, as reported in ACE log files (or the Tcl Console) during a prior flow execution
  A = the total available (unused) RAM memory,
  A = *R* – (*O* + *G* + *B* + *U*), where
  R = total RAM installed in the workstation
  O = amount of memory required by the Operating System
  G = amount of memory required by the currently-running ACE GUI
  B = amount of memory required by the currently-running ACE backend process (named `acx` or `acx.exe` in process lists)
  U = amount of memory required by all other user processes expected to execute while the background processes are running

Continuing the example of the 8 thread 32GB workstation: If the workstation is running Linux, estimate the OS requires 0.5GB, the ACE GUI process requires 1GB, the GUI backend process (`acx`) requires 3GB, and no other user processes are running; the available memory *A* = (32GB – (0.5GB + 1GB + 3GB + 0GB)) = 27.5GB. If the log files of a prior run report the user design requiring a peak memory usage of 7GB, then the memory constraint value is (27.5GB / 7GB) ≈ 3.9. The processor constraint would be (8 threads + 1) = 9. The lesser of the two values is the 3.9 for the memory constraint. So following the guidelines, the ideal parallel process count would be between **3** and **4**. To completely balance the two constraints for the design, the example user would need 7GB × 9 threads = 63GB of available memory before they could expect optimal performance running 9 parallel processes.

Tip

ACE Memory Utilization

ACE logs the amount of memory (RAM) used by the backend as a design proceeds through the flow. This number is reported at the end of every flow step (see page 221) in the log files and (when the GUI is running the flow in single process mode) in the Tcl Console. It is also possible to directly query ACE at any time to find out the peak backend memory usage in KB with the `get_ace_peak_memory_usage` Tcl command. These features should allow an educated decision to be made as to how much memory each parallel background process requires for the design, and thus how many processes may be executed in parallel within the current memory constraints.

Example from log

```
Flow step "report_timing_final" completed in 1 seconds. Peak memory usage is 4917 MB.
```

Example from Tcl Console View query showing peak memory use in KB

```
cmd> get_ace_peak_memory_usage
5035008
```

Configuring ACE to Use an External Job Submission System

Due to the wide variety of grid, batch, queue and cloud job submission systems available, it is not possible for ACE to support each individual product specifically. Instead ACE Multiprocess can be configured to interface with whatever job submission system is available at the user site.

Minimum Requirements

Currently, the following are required for the minimum functionality:

- The name of the job submission executable or script (providing a full directory path to the executable or script is recommended, though it may not be necessary in some PATH configurations).
- The job must be submitted in synchronous/blocking mode (the job submission process must not complete until the ACE child process/job has completed execution). ACE itself currently has no support for the tracking of job status through periodic queries as would be necessary with asynchronous/non-blocking jobs.

> **Warning!**
>
> A non-blocking/asynchronous job submission system currently risks data corruption, because ACE can no longer guarantee it knows when the job is complete, so ACE cannot properly manage data locking states across the simultaneously executing implementations.

- An exit code of zero from the job submission process indicates success.
- A non-zero exit code from the job submission process indicates failure. The Multiprocess system simply reports success/failure based upon the exit code value.

Presently, if the job system at the user site is not already a synchronous/blocking system, then it is necessary for the user to write their own script or executable which approximates synchronous/blocking functionality.

In theory, this should be possible:

1. Submit the job.
2. Capture the unique identifier for that job.
3. Loop while querying the job status (using the previously captured unique job identifier from the job system) until completion is indicated.
4. Capture the job exit code.
5. Return the appropriate exit code (to ACE) indicating the success or failure of the job status.

After the job submission request completes, and after any network files have been written, the ACE Multiprocess GUI reads the output files from the submitted job, gathering the information needed for the Multiprocess Summary Report. The read of the result files only happens once per job.

If the user job submission process finishes before the submitted ACE job is complete (as would happen with a non-blocking job submission system), the ACE implementation output files are either missing or incomplete when queried, and the Multiprocess Summary Report shows that no results were found for that ACE job.

Optional Improvements

When external job submission systems are properly configured, the following features are also available within ACE Multiprocess:

- Support for killing or cancelling submitted jobs
- Assignment of the job working directory
- Assignment of a job name
- Streaming real-time log output for each Job

Killing or Cancelling Already-submitted Jobs

For simplicity, the ACE Multiprocess system only manages the job through the (blocking) job submission process. The Multiprocess system currently does not track job identifiers or any special job status logged by the job submission process itself. When ACE needs to cancel or kill the job, it essentially sends a "kill" (technically a "SIGINT" in Linux) to the (blocking) job submission process. It is expected that this also kills/cancels the underlying ACE job. If this does not actually kill the underlying ACE job (or remove it from the appropriate job queue, etc.), then it becomes the responsibility of the ACE user to manually kill the job with the job submission tool.

Job Working Directory

In some cases it might be necessary to specify the working directory of the ACE job as a command line argument to the job submission process. While ACE jobs lacking an explicit working directory assignment are known to run without errors in most situations, some job submission systems may require the explicit assignment of a working directory. The working directory specified by ACE for a job changes for each implementation and, typically, is the implementation directory itself.

Job Name

It is extremely convenient for ACE Multiprocess to have a way to pass in the job name as a command line argument to the job submission process.

The job name does not aid ACE directly, but is intended to assist external users of the job submission system in tracking job status, job lifetime, queue management, etc. through other (non-ACE) tools.

The job name is currently made unique by concatenating the following information, with items in brackets replaced by their logical values:

- `ACE_Multiprocess_[user name]_[project name]_[implementation name]`

Additionally, special characters found in the variable values are replaced by the underscore (_) character.

Streaming Real-Time Job Log Output

ACE logs all of its normal output in a log file, which gets post-processed after job completion to verify how far ACE went through the flow, and to harvest the reported timing information for inclusion in the Multiprocess Summary Report. However, properly configuring the following can help the user track the progress of the ACE jobs as they run.

If the job submission process redirects or pipes the standard output and standard error streams from the underlying ACE job, so that the job submission process re-transmits that same data on its own standard output and standard error streams, then ACE may be able to show the streamed job output during the Multiprocess run.

If the underlying ACE job standard output and standard error streams are redirected to a file, preferably through user-managed command line options for the job submission process itself, then ACE may be able to show the streamed job output from the file during the Multiprocess run.

> **Note**
>
> Due to various concerns such as network file write caching and the occasional complexity of shell redirection in spawned processes, this job submission log file option may be difficult to get working properly.

Configuring ACE

The external job submissions are performed via a user-configurable command-line executable. The configuration is managed through the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201), reached by following the **(configured in Preferences)** hyperlink in the Multiprocess View. As a potentially useful example, by default ACE is configured to use GridEngine (see http://en.wikipedia.org/wiki/Oracle_Grid_Engine) through the `qsub` command. (When using a system other than the GridEngine, users need to clear all fields on that preference page and provide the values which are appropriate for their own system.) For the configuration to work, the job submission command must be in the path (or have its path fully specified), and the ACE executable must be reachable from the job system execution hosts.

ACE is able to optionally provide some values to the job submission system if the related argument fields are populated. The optional values ACE may provide are:

- The working directory for the ACE Multiprocess job
- The job name
- The path and filename to be used by the job submission log file

It is extremely likely that additional command-line arguments are required by the job submission executable in order to meet the ACE minimum requirements. Additional arguments are also typically needed to assign execution queues, memory limits, etc. These additional arguments (and any argument values) should be specified on the preference page as well.

Caution!

Command-line arguments must not be specified in the **Job Submission Executable** field. Attempts to do so fail.

Debugging Job Submission System Configurations:

If the job submission system is properly configured on the host machine, (meaning it is possible to successfully execute non-ACE tasks using the job submission executable from the command line), and ACE is still unable to successfully submit jobs to the system, please contact Achronix technical support.

Warning!

Potential for File Corruption

Attempting to manually run the logged command on the command line (without the Multiprocess View additional automated safety locks in place) might cause ACE data file corruption.

While ACE does provide the complete attempted job submission command in the "Multiprocess Run Logs" section of the Multiprocess view, *DO NOT* copy the text of the attempted command and manually attempt execution from the command line. A large number of assumptions are made (including bypassing the normal project-level and implementation-level safety checks which prohibit file corruption) when ACE is executed using the provided command options and Tcl batch script — these assumptions are violated during manual execution attempts.

Network File System Latency Concerns

When dealing with external job submission systems, network drive latency becomes a concern. The ACE multiprocess system waits for each external process to complete before it harvests the timing information for that implementation. To avoid potential hangs (where the multiprocess system mistakenly waits forever for a file to appear, or for a file to be completely written), there is a configurable timeout setting, which is by default 5 seconds. If, after the external process for an implementation has completed, the timing summary information cannot be found within the allowed number of seconds, then the Multiprocess Summary Report (see page 238) shows the message "No Timing Results Found" for that implementation.

Note

A "No Timing Results Found" message for an implementation in the summary report means the timing information needed for the summary was not available within the allotted time. The allotted time may be increased using the **Allowed seconds of NFS write latency** setting in the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201), as shown in the image below.

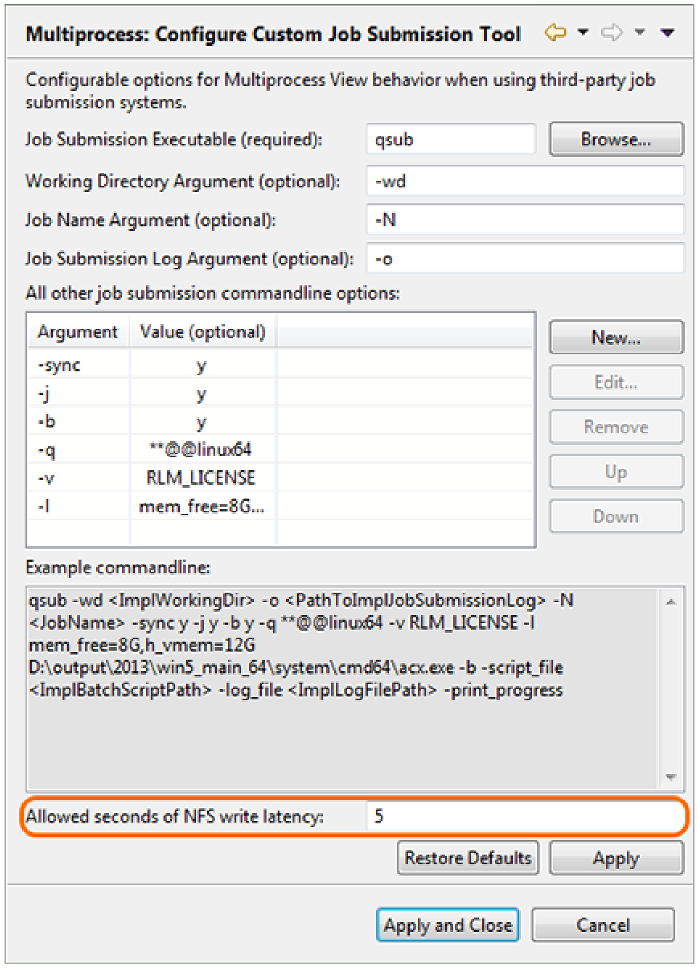


Figure 130: ***Configure Custom Job Submission Tool Dialog***

Configuring the Desired Flow to be Followed by the Selected Implementations

All the implementations run through the Multiprocess View follow the same flow steps (see page 221) through the flow (see page 221), as configured in the Flow View (see page 61). Thus, ensure that all optional flow steps are enabled /disabled as desired before starting multiprocess execution.

Additionally, in the section of the Multiprocess View labeled "Multiprocess Flow Management (see page 86)", it may be chosen to stop the multiprocess flows early, prior to traditional "completion". For example, when designs are known to be incomplete, and thus known to fail the **Run Final DRC Checks** flow step, it may be chosen to stop the flow prior to running that flow step.

To stop all the multiprocess flows at a given flow step, simply select that flow step in the **Stop Flow After:** drop-down list. No subsequent flow steps are executed for the selected multiprocess implementations.

As a convenience, since optional flow steps are frequently chosen to be the final multiprocess flow step, there is a **Force Selected Flow Step to be Enabled** checkbox. When checked, if the selected final flow step is optional and not enabled, then as the multiprocess implementations are scheduled, the selected flow step is enabled for all the multiprocess implementations before they begin execution. If this checkbox is left unchecked, and a disabled optional flow step is selected as the final step, then the final step executed is the last enabled flow step prior to the selected step.

For example, if **Stop Flow After** is set to **Run Post-Route Timing Analysis** (an optional step), but this flow step is disabled in the Flow View, and if **Force Selected Flow Step to be Enabled** is not checked, then the multiprocess flows stop after the (required) **Run Route** flow step, since that is the last enabled step prior to **Run Post-Route Timing Analysis**.

Selecting the Implementations to be Run in Parallel

To select the implementations to be run in parallel:

1. In the Projects View (see page 125), select the desired project (see page 215). The Implementation Table within the Multiprocess view Select Implementations (see page 86) section is updated to display data for the active project and implementation (see page 221).
2. In the Multiprocess View, ensure the **Existing Implementations** radio button within the "Select Implementations" section is selected. This limits the contents of the Implementation Table to just the implementations which already exist for the active project (generating and executing new implementations using option sets (see page 215) is covered in Attempting Likely Optimizations Using Option Sets (see page 352)).
3. In the Implementation Table, all listed implementations are selected (the checkbox in the Implementation column will be checked) by default. Implementations may be selected/deselected in bulk with the **Select All** and **Deselect All** buttons. Individual implementations may have their selection toggled by clicking their checkboxes in the first column of the Implementation Table.

> **Tip**
>
> If the implementation table is not large enough (or is too large) for the full implementation list, simply collapse and/or expand one of the other sections in this view (left-click the section title). This causes the table to resize to exactly fit the current implementation list.

Starting Background Execution

To start background execution:

1. When the parallel count has been set, the flow has been configured, and the desired implementations have been selected, click the **Start Selected** button, or the equivalent ( ) **Start Background Queue Execution** action in the Multiprocess view local button-bar or menu, to begin background multiprocess execution.
2. After multiprocess execution has been started, the **Parallel Queue Count** and Implementation Table is disabled. They are not re-enabled until multiprocess execution is completed. In the Multiprocess Run Logs (see page 88) section, a new tab is created for the logged output of each selected implementation. The log info in each tab is updated live as the corresponding implementation process executes (the displayed log info mirrors the information captured in the log files (see page 218) for each implementation).
3. As implementations are queued, start execution, and complete execution, the implementations execution states (see page 88) are updated in the implementation table, and each implementation log tab icon is also updated to show the current execution state.

> **Note**
>
> Presently, it is not possible to control the order of implementation execution.

Caution!

For safety, all ACE Tcl commands (i.e., most ACE GUI interactions) are blocked while multiprocess execution is underway. Blocked Tcl commands are queued and allowed to run when multiprocess execution is completed. Similarly, multiprocess execution is blocked until all in-process and already-queued ACE Tcl commands (including running the Flow in the foreground) are completed.

Stopping/Canceling Background Execution

All queued and executing background implementations may be quickly cancelled by selecting the **Stop All** button below the Implementation Table, or the equivalent ( ) **Stop All Background Queue Execution** action in the Multiprocess view local button bar or menu.

It is also possible to cancel execution of individual implementations. This may only be done via the ( ) Progress View. During multiprocess execution, a ( ) button to show this view is visible in the lower-right of the ACE status bar. This view is also available by selecting **Window** → **Show View** → **Other...** → **General** → **Progress**. The Progress View displays all queued and currently-executing background tasks, including the tasks for the background implementation processes. To the right of each listed incomplete background task is a ( ) stop icon, which cancels/stops execution of that task. Because the Progress View can list more tasks than just the background multiprocess implementations, caution should be used to avoid cancelling/stopping the wrong task.

Viewing the Results

After the first implementation completes execution, an HTML Multiprocess Summary Report (see page 238) file is created and opened in ACE (the report file is created in the project directory, and is named `multiprocess_summary.html`. This action automatically overwrites previous multiprocess summary reports without prompting). As each subsequent implementation completes execution, the multiprocess summary report is updated with the latest data.

As implementations complete execution, their execution states (see page 88) change appropriately. If an implementation encounters errors while running the flow, the execution state for that implementation becomes the Error state, which is reflected by the icon shown both in the log tab and the Implementation Table. In addition, the tooltip for the appropriate log tab and Implementation Table entry is updated to include a summary of the captured error messages. Error details are visible in the log messages shown in the tab, as well as within the Implementation Log (see page 219) and Multiprocess Log (see page 219) for that implementation.

Caution!

There is a known sequence whereby all multiprocess results are identical. If there is an existing project with previously generated option sets, and ACE has been upgraded to a newer version, it prompts when opening the existing project to reset the implementation (see page 215) options to the defaults for the new version of ACE. The recommendation is to accept this reset as a new version of ACE may include new implementation options which are only applied by accepting this reset. At the same time, older implementation options that have been deprecated are removed.

The issue is that currently ACE resets all of the option sets (see page 215) to the same default values. Subsequently, when a multiprocess flow is run with the new project, all results are identical. The workaround is, after having upgraded ACE to the new version and accepting the implementation (see page 215) option reset, delete all the implementations other than the original base implementation and then regenerate the option sets (see page 215).

Multiprocess Batch Mode

Overview

To obtain the highest QoR, ACE supports running multiple different implementations in parallel using Multiprocess. Multiprocess is available from the ACE GUI and is described in:

- Multiprocess View (see page 83)
- Running Multiple Flows in Parallel (see page 282)
- Attempting Likely Optimizations Using Option Sets (see page 352)

Multiprocess batch mode provides the same functionality as the GUI, but can be run from the ACE Tcl console command line or by using an external Tcl script. The relevant Tcl command is run_multiprocess (see page 587).

Modes

Similar to running Multiprocess from the GUI, Mutliprocess batch mode must be run in the context of a currently Active Project and Implementation (see page 221). The current active project is used as the basis for all the implementations that are run, with the current active implementation used as the basis for any newly-generated implementations.

Multiprocess batch mode supports three modes of operation:

1. Generate implementations from option sets (default setting).
2. Seed sweep (`-seed_sweep`).
3. Use existing implementations (`-use_existing_impls`).

A full list of all options is given in the run_multiprocess (see page 587) manual page.

Generate Implementations From Option Sets

Running from option sets is the default mode of operation for Multiprocess batch mode and is used when neither `-use_seeds` nor `-use_existing_impls` is specified. This mode generates fresh implementations for every available option set definition. Previously existing implementations with the same name are overwritten. See Attempting Likely Optimizations Using Option Sets (see page 352) for additional details.

The currently active implementation is always included as one of the executed flows when this mode is used.

Seed Sweep

The seed sweep mode generates fresh implementations (based upon the active implementation) for every specified seed value. Previously existing implementations with the same name are overwritten.

Seed sweep mode is selected by use of the `-use_seeds` argument:

```
run_multiprocess -use_seeds {5 7 13}
```

The current active project and implementation forms the basis of each generated implementation, with the implementation option "seed" set to the given seed value as an override of the seed inherited from the active implementation.

Implementations created during seed sweep are named `{active_impl_name}_seed{value}` (e.g., `impl_1_seed18` for a seed value of 18 and an active implementation name of `impl_1`).

The currently active implementation is always included as one of the executed flows when this mode is used.

Use Existing Implementations

The Use Existing Implementations mode does not generate any new implementations, but simply runs each of the named implementations. Use Existing Implementations mode is selected by use of the `-use_existing_impls` argument:

```
run_multiprocess -use_existing_impls {impl_1 impl_1_improved impl_1_experimental}
```

To run all existing implementations, specify:

```
run_multiprocess -use_existing_impls [get_impl_names]
```

Unlike the other modes, the currently active implementation is *NOT* included as one of the flows run unless it is explicitly named in the `-use_existing_impls` list.

Flow Steps

An important principle to understand is that the enabled or disabled flow steps of the currently active implementation are inherited by all implementations executed during Multiprocess batch mode. Therefore, before commencing the Multiprocess batch mode, the user should ensure that the currently active implementation has the desired flow steps enabled.

In addition, Multiprocess batch mode can be configured to stop at an explicit step via the `- stop_flow_at` argument. This argument can be used to terminate each flow at a particular step. For example, if `report_timing_routed` is specified, then none of the DRC or bitstream flow steps are performed. Using this argument reduces the overall time taken for Multiprocess batch mode, as each implementation runs a reduced number of flow steps. After Multiprocess batch mode has completed and an implementation found which achieves the desired QoR, then that implementation can be loaded into ACE, and the final flow steps executed. With the aforementioned example, which was stopped at `report_timing_routed`, the routed `.acxdb` can be loaded into ACE, and the DRC and bitstream generation flow steps executed to produced the required bitstream.

> **Note**
>
> If the flow step specified by `- stop_flow_at` is disabled when the multiprocess run begins, it is explicitly enabled.

Getting Started

The commands to start Multiprocess batch mode vary according to whether ACE is running in command-line mode, batch mode (using a script file), or from within the ACE GUI.

Command-line Mode (Interactive)

1. Open ACE in command line mode (`ace -b`). See Running ACE (see page 259).
2. Use restore_project (see page 584) to load the project.
3. Set the active implementation (see page 600).
4. (Optional) Use disable_flow_step (see page 545) and enable_flow_step (see page 550) to configure any flow steps desired/needed or bypassed for all of the implementations that are to be run.
5. Issue run_multiprocess (see page 587) command. See examples (see page 295) below.

Batch Mode (Script File)

1. Open ACE in command-line mode, passing in a script file. Use script arguments to specify the project name (`ace -b -script_file <my_mp_batch_script.tcl>`). See Running ACE (see page 259).

Code

```
$ ace -batch -script_file <my_mp_script> -script_args <my_project_name>
```

An example script file, using the project names as the first argument is shown below.

Code

```
# Script file to run Multiprocess batch mode
set my_proj [lindex $argv 0]

# 1. Restore the project
restore_project $my_proj

# 2. Set active implmentation (to the default)
set_active_impl impl_1

# 3. (Optional) Ensure Run Estimated Timing Analysis flow step is enabled. Disable Generate Bitstream
enable_flow_step report_timing_routed
disable_flow_step write_bitstream

# 4. Run Multiprocess batch mode generating new implementations from option sets
#     Set to a maximum of 8 jobs
#     Stop after Post-Route Timing Analysis.
run_multiprocess -parallel_job_count 8 -stop_flow_at report_timing_routed
```

> **Note**
>
> When running in the command-line mode, or batch mode, in order to cancel a Multiprocess batch mode run, CTRL+C must be used.

ACE GUI

1. Open ACE GUI.
2. Load the project (see page 270).
3. Set the active implementation (see page 278).
4. (Optional) Using the Flow View (see page 61), select or deselect any flow steps desired/needed or bypassed for all of the implementations that are to be run.
5. In the Tcl console window, issue the run_multiprocess (see page 587) command. See examples (see page 295) below.

Examples

Running all Option Sets

To run all option sets, using the existing maximum job count as specified in your ACE GUI preferences:

```
run_multiprocess
```

To run all option sets, limiting concurrent jobs to 8:

```
run_multiprocess -parallel_job_count 8
```

Running a Seed Sweep

To run a seed sweep using preferred seed values:

```
run_multiprocess -use_seeds {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31}
```

Re-running Four Existing Implementations

To re-run four existing implementations all at the same time:

```
run_multiprocess -use_existing_impls {impl_1 impl_1_acx_mux_utl_seed impl_1_acx_seed21
impl_1_acx_seed33} -parallel_job_count 4
```

Re-running an Existing Implementation

To re-run an existing implementation, stopping just after running "write netlist final":

```
run_multiprocess -use_existing_impls {impl_1_seed88} -stop_flow_at write_netlist_final
```

Running all Option Sets

To run all option sets on the grid, with custom job submission parameters:

```
run_multiprocess -use_job_submission 1 -jobs_wd my_jobs_working_dir -jobs_name my_job_name -jobs_log
my_jobs_logfile -jobs_args {{-sync y} {-j y} {-b y}}
```

Progress Monitoring

Within the Tcl console or shell, `run_multiprocess` checks each of the input arguments to ensure they are correct (including checking that any specified existing implementations exist) and then launches the requested number of parallel implementations runs. Within the Tcl console or shell, `run_multiprocess` indicates the start and completion (success or failure) of each implementation run.

To monitor progress, use the Multiprocess Summary Report (see page 238) (`multiprocess_summary_report.html`) file that indicates which implementations have completed and their timing summary. This report is generated regardless of whether the Multiprocess batch mode was run from a command shell, or from within the ACE Tcl console. This report can either be viewed external to ACE using a web browser, or within ACE as detailed below.

Viewing Multiprocess Summary Report within ACE

Open the Multiprocess Summary Report from the Projects view, by right-clicking the project.

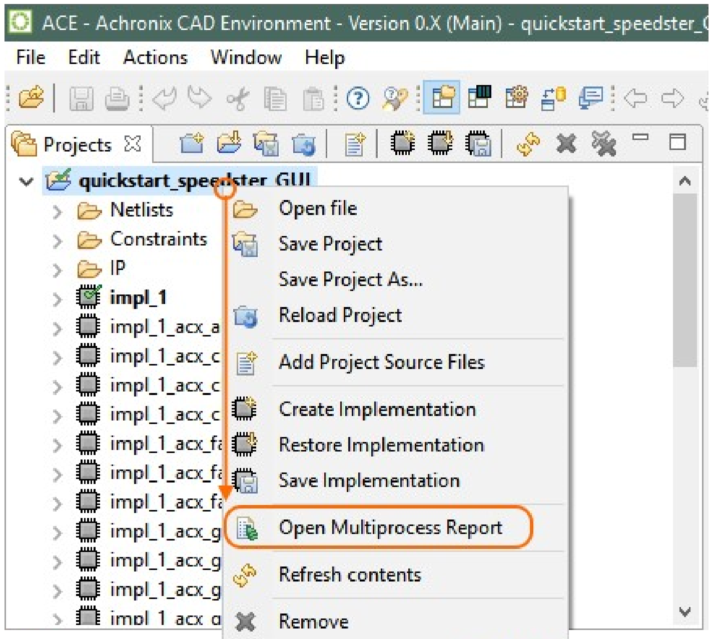


Figure 131: *Open Multiprocess Report*

The report can be refreshed by right-clicking in the view and selecting **Refresh**, or by pressing the refresh hotkey, **F5**, when the report tab has focus (click the report first). The report view does not automatically update when using Multiprocess batch mode. This behavior differs from when Multiprocess is run directly from the GUI, where the report view is automatically updated.

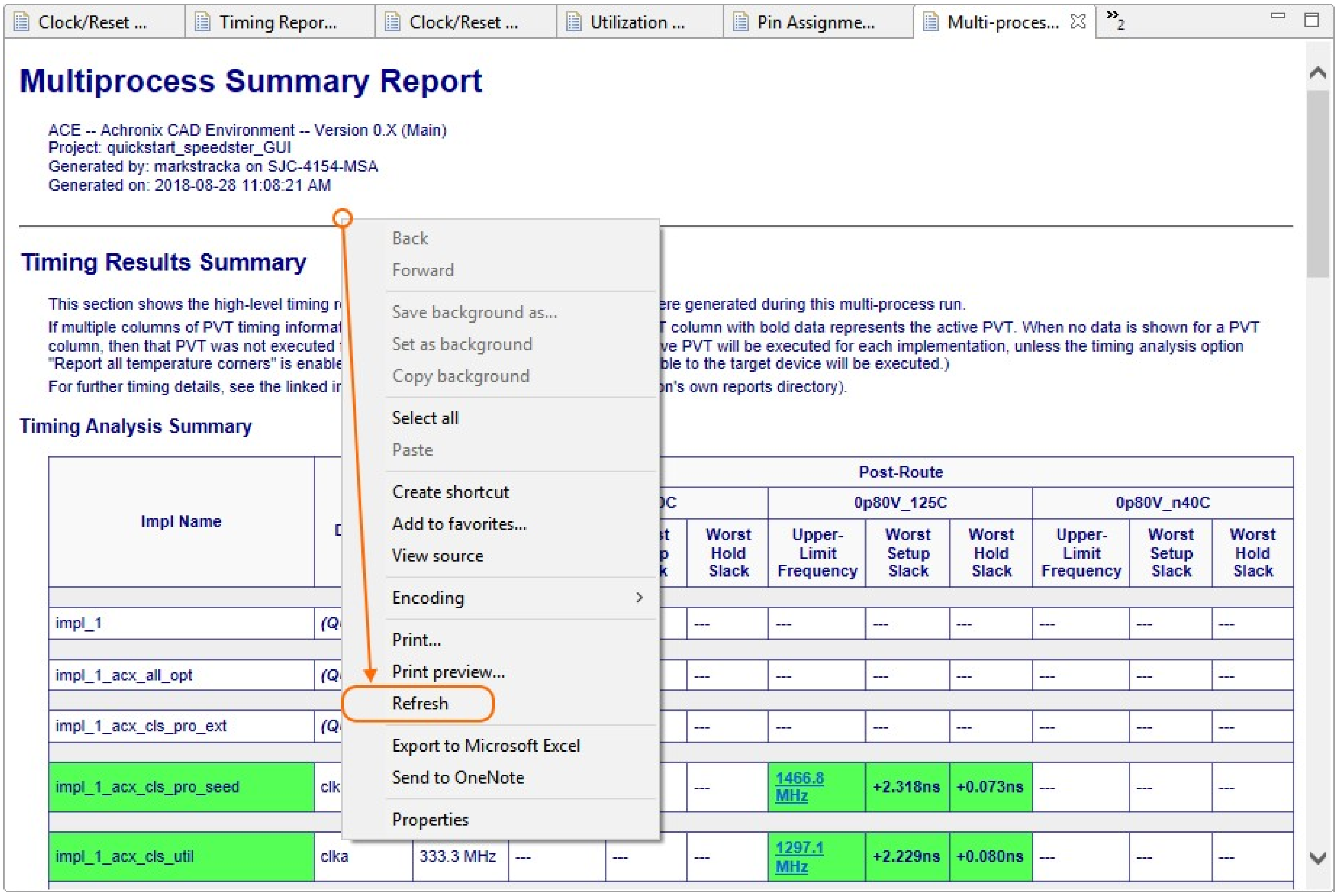


Figure 132: *Refresh Multiprocess Report View*

Furthermore, progress monitoring can be achieved if a job submission system is used — completed jobs might be able to be monitored using the job submission system tool suite. Finally, to see the status of an individual implementation, open the `<implementation>/log/multiprocessImpl.log` file, and monitor updates to the individual implementation progress.

Stopping the Running Implementations

When running in command-line or batch mode, use CTRL+C to cancel a Multiprocess batch mode run. When running in the ACE GUI Tcl console, use either CTRL+C to cancel Multiprocess batch mode or use the Progress View to cancel the process.

Detecting Changes to Project Source Files

ACE provides a rich set of features to enable detecting changes to project source files against the state of the project files loaded into the ACE database during the Run Prepare flow step, as described in the following sections.

Files Open in the ACE Editor Area

If a project source file is open in a text editor window, a pop-up dialog box appears offering to refresh the contents of the stale file in the ACE editor tab if the source file is changed on disk.

Smart Change Detection Using Custom Checksums

Instead of caching timestamps for files to perform the checking, ACE caches custom file checksum values. The checksums are computed in a robust way that ignores comment lines and whitespace lines, so that only the actual Verilog netlist or SDC/PDC constraints commands are used in the checksum. This allows generated files, such as the Synplify netlist, to be regenerated from the same RTL which produces the same gate level netlist, but with different comments at the top, to be treated as an unchanged file. Timestamps are inherently fragile and change when a project is copied from one directory to another. This checksum approach is much more robust and does not flag a source file as changed unless its content is meaningfully changed.

Saving the Active Implementation

When ACE saves an ACXDB file (when the state of the ACE database is saved for the active implementation), it caches the checksums of all project source files used to create that state in the DB. The project source file checksums are saved inside the `.acxdb` file.

Restoring the Active Implementation

When ACE loads/restores an ACXDB file (when saved place and route data is loaded from the `.acxdb` file on disk into the ACE database for the active implementation), ACE checks all project source files and checksums on disk against the cached project source files and checksums inside the ACXDB file. If a project source file has been added to the current ACE project, removed from the current ACE project, or if its file checksum has changed, ACE prints a warning message to the Tcl Console and ACE log file. This alerts users that the saved ACXDB is out of sync with the current project source files.

Caching the Project Source File State

The **Run Prepare** flow step is the first flow step, and is where all project source files are loaded into the ACE database from disk. Whenever the **Run Prepare** flow step is run, ACE caches the project source files and checksums for all files used in the active ACE project implementation.

Automatic Checking while Running the Flow

Each flow step (**Run Place**, **Run Route**, Timing Analysis, Final DRC checks, etc.) checks the project source files (including the `.acxprj` ACE project file itself) and checksums on disk against the files and checksums cached at the beginning of **Run Prepare**. If any file is missing, added, or out of sync, ACE reports a warning at the end of each flow step to the Tcl Console and ACE log file:

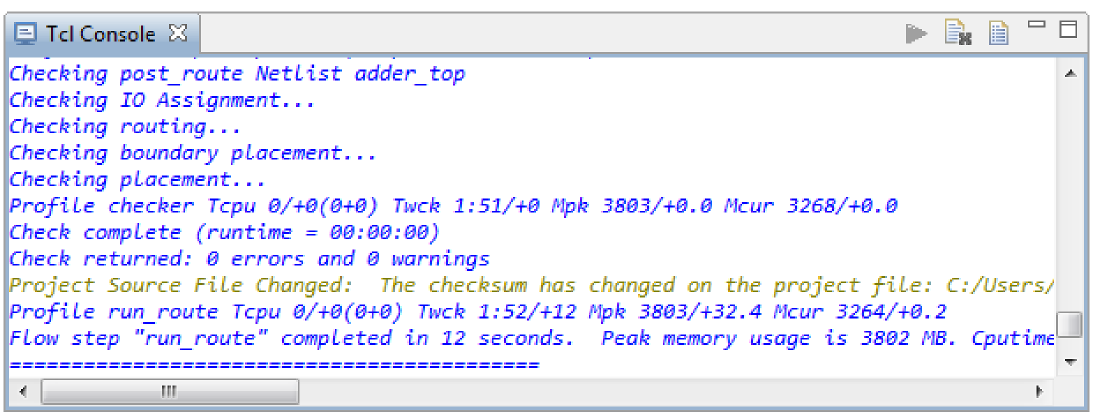


Figure 133: ***Missing, Added, or Out of Sync File Warning Example***

Warning Visualization in the Flow View

If any flow step reports a warning about out-of-sync files, all completed flow steps in the Flow View (see page 61) are marked with a yellow warning icon instead of the green checkmark icon to indicate that the step is complete, but is out of sync with the source files on disk. The tooltip text in the Flow View shows all the warning messages.

Even when no flow step Tcl commands are running, the GUI checks the project source files and checksums every 5 seconds (by default) in a background thread. If any file becomes out-of-sync, all completed flow steps in the Flow View are marked with a yellow warning icon instead of the green checkmark icon to indicate that the step is complete, but is out of sync with the source files on disk.

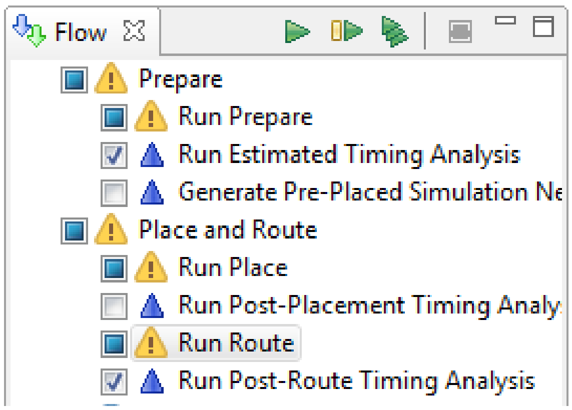


Figure 134: ***Out of Sync File Warning Example***

Pop-up Dialog Warnings

If the flow is running, a "Project Source Files Changed" dialog appears if a change to project source files is detected during the built-in check at the end of each flow step. The same warning messages that are printed to the Tcl console are displayed in the dialog. Optionally choose to cancel running the rest of the flow, or let the flow continue. The flow continues to run in the background until the choice is made. The pop-up dialog has a preference checkbox which allows disabling the pop-up from being shown in the future.

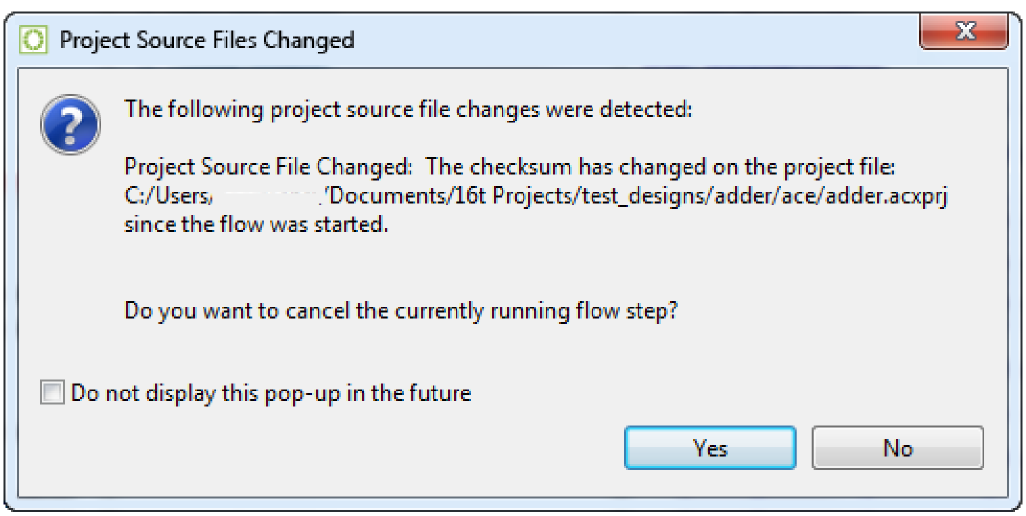


Figure 135: ***Project Source Files Changed Dialog Example***

If the flow is not running, a different “Project Source Files Changed” dialog appears if a change to project source files is detected during the built-in check at the end of each flow step. The same warning messages that are printed to the Tcl console are displayed in the dialog. There is no choice for cancelling the running flow, since the flow is not running. The pop-up dialog has a preference checkbox which allows disabling the pop-up from being shown in the future.

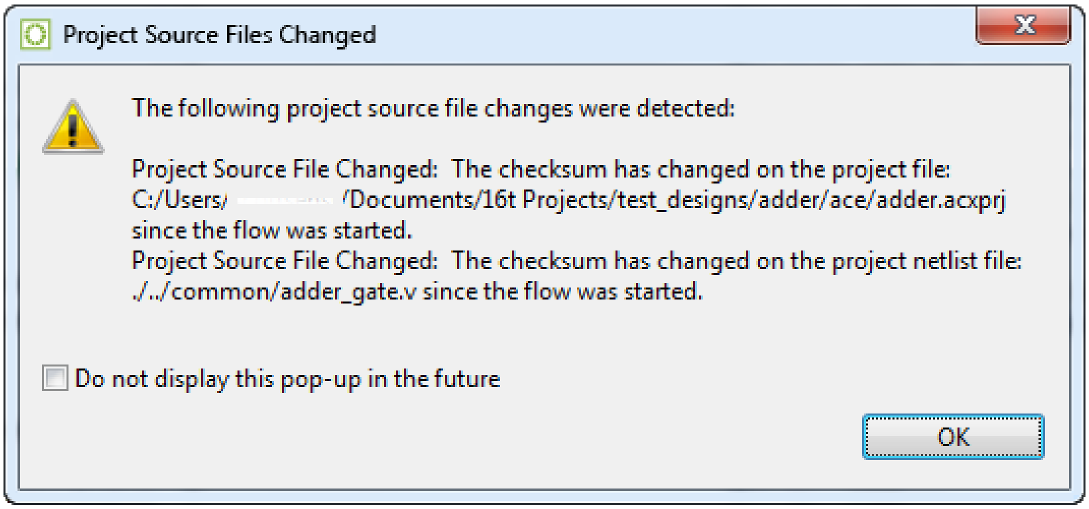


Figure 136: ***Project Source Files Changed Dialog Example When Flow Is Stopped***

> **Note**
>
> **Minimal Pop-up Interruptions**
>
> In general, pop-ups are minimized to only alert new changes. The Project Source Files Changed dialog appears only when a new change is detected. So if the gate level netlist file is changed while the flow is running, the pop-up (if enabled by the user preference) appears. If the same gate level netlist file is then changed several more times, no further pop-up appears since there has already been a notification that the file is different than the original source file. However, if a project constraints file (in addition to the gate level netlist) is then changed, the Project Source Files Changed dialog appears again to alert that now 2 source files are changed.

Managing Pop-up Preferences

There is a user preference to enable/disable the Project Source File Changed pop-up at the bottom of the Project Management Preference Page (see page 210). This preference setting is the same as that controlled with a checkbox in the Project Source File Changed Dialog. From the main menu bar, select **Window** → **Preferences** and select **Project Management** on the left-hand side of the Preferences dialog. This preference page can be used to re-enable the pop-up if the checkbox in the dialog is enabled.

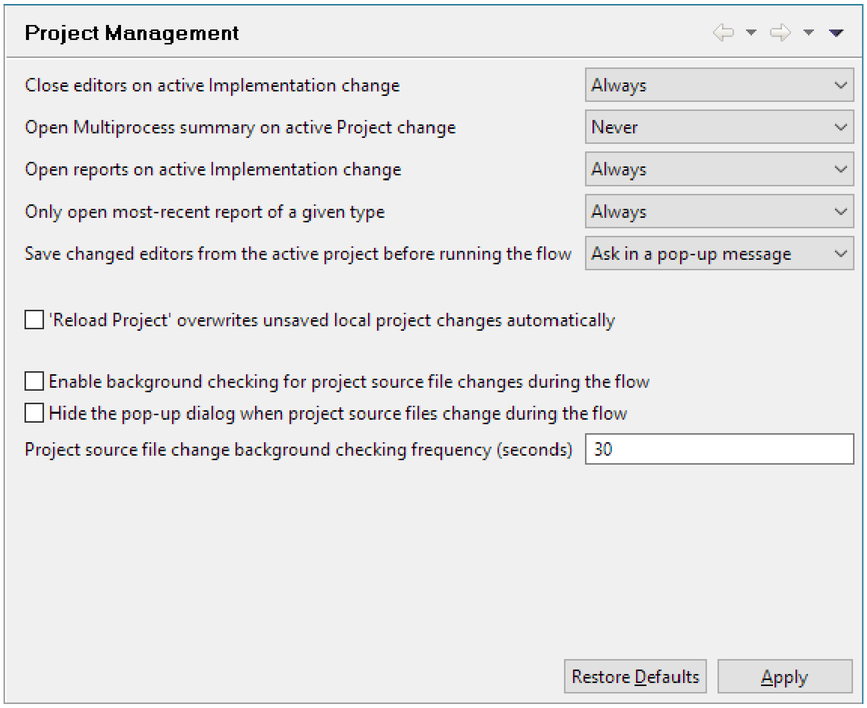


Figure 137: ***Project Management Preference Page Example***

Tcl Command Support

The `check_project_status` Tcl command can be called to manually check file and checksum consistency outside of the built-in checks performed at the end of each flow step. If any file is missing, added, or out of sync, ACE reports a warning to the Tcl Console and ACE log file. This command only applies if the flow has run at least through the **Run Prepare** flow step and there is a design loaded in the ACE DB.

Using the Tcl Console

Any operation that changes project or design data can be performed from the command line via a Tcl command. The Tcl Console view (see page 144) provides an interface from within the GUI for viewing and executing Tcl commands.

Sending Commands from GUI Actions

Any action in the GUI that changes project or design data automatically sends a Tcl command through the Tcl Console view (see page 144) to do the work. All Tcl commands generated by GUI actions are displayed in the Tcl console along with any output from the command.

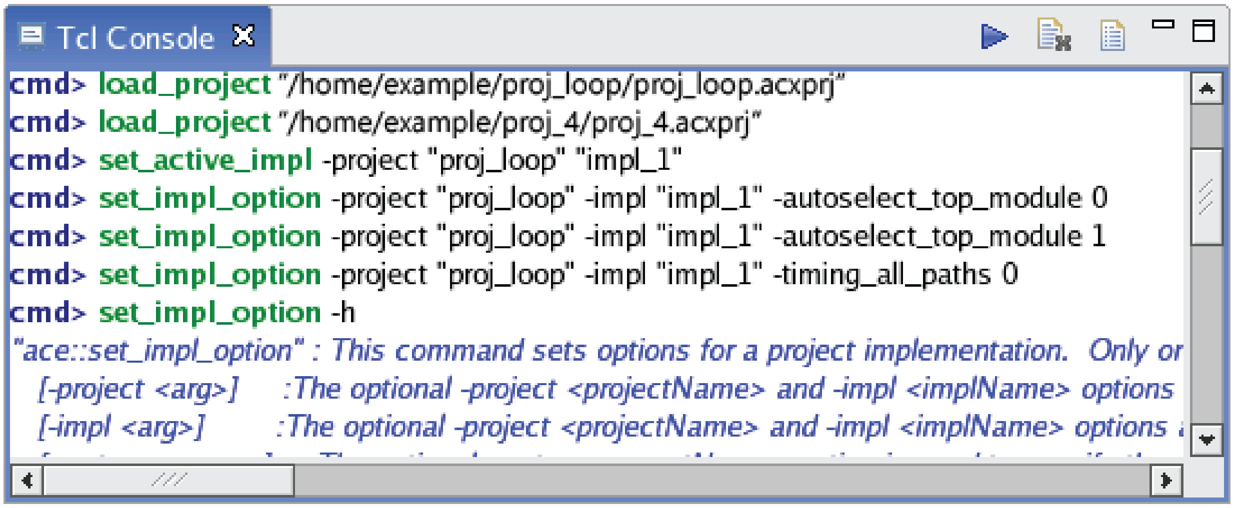


Figure 138: *Tcl Console Example*

Sending Commands from the Console

To send a command from the Tcl console:

1. Enter or paste the command text at the available **cmd>** prompt in the Tcl console view. Valid commands are highlighted in bold green.
2. Either press **ENTER** or click the ( ) **Send Command** toolbar button in the Tcl console view.

All output from the command is displayed in the Tcl Console view under the command prompt. Informational messages are displayed in italic blue text. Warning messages are displayed in italic yellow text. Error messages are displayed in italic red text.

Command Highlighting

Text entered in the Tcl console is checked against the valid set of user Tcl commands. Valid commands are highlighted in bold green.

Command Auto-Completion

When typing into the Tcl console, pressing the **TAB** key pops up a Tcl command auto-completion dialog if the current cursor position in the text has any possible matches. If no possible matches are found, an error beep sounds.

Pressing the **TAB** key at an empty **cmd>** prompt pops up the full list of available commands. When the command auto-completion dialog is open, use the arrow keys to navigate up and down the list of choices and press the **Enter** key on a selected command to complete it at the command prompt. Typing while the command auto-completion dialog is open shortens or lengthens the list of valid commands, depending on the cursor position in the Tcl console view.

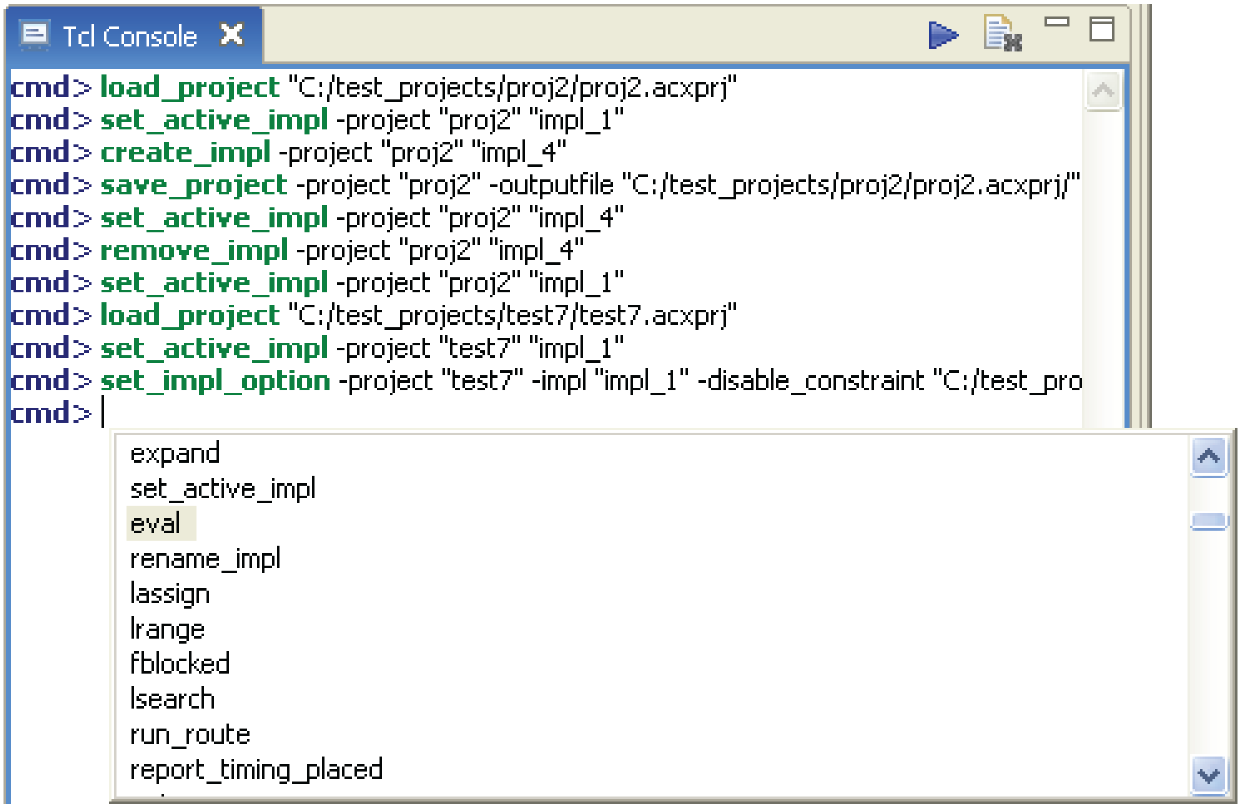


Figure 139: *Tcl Command Auto Completion Dialog Example*

Command Help

When the command auto-completion dialog is open, help text appears to the right of the command list for the selected command.

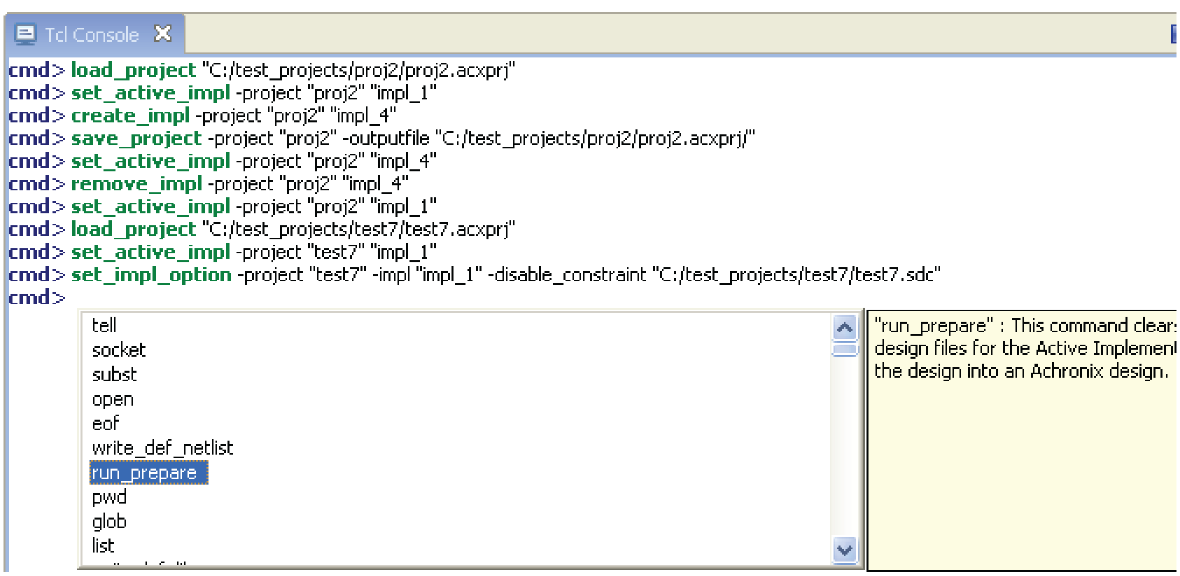


Figure 140: *Tcl Command Auto Completion Dialog Help Text Example*

To view help text for commands, either bring up the command auto-completion dialog and select the desired command, or enter the command name at the **cmd>** prompt and use the `-help` argument to output the help text to the Tcl console.

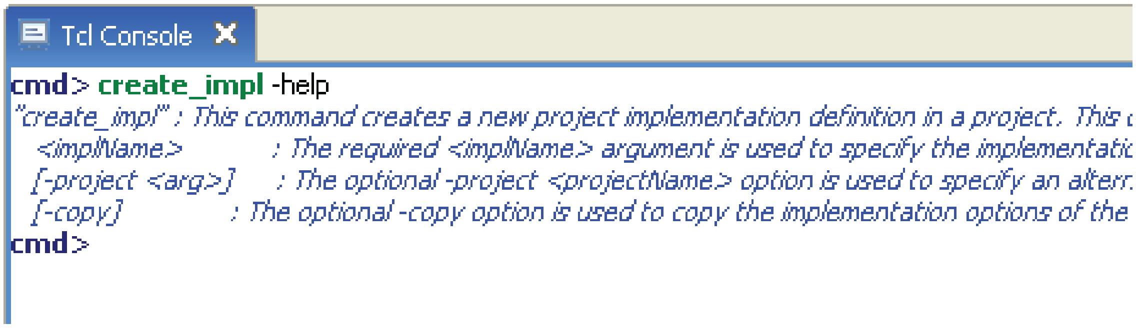


Figure 141: ***Tcl Command Help Option Example***

Text Limit

The Tcl console view has a limit of 2000 lines. When this limit is reached, any new lines entered via commands or message text causes the text at the top of the Tcl Console to be pruned.

Additionally, when Tcl command return values are displayed in the Tcl console, any long returned values are visually truncated at 500 characters in the console. The actual returned value is not edited, just the textual representation shown in the console.

Note

- Performance of the Tcl console is much higher when there are fewer than 2,000 lines of text.
- Hitting the text limit does not clear the contents of the ACE log file. All messages continue to be logged in the log file and earlier messages are not removed.

Clearing the Console

Text in the Tcl console view can be cleared by clicking the ( ) **Clear Console** toolbar button in the Tcl console view. This action truncates all text in the console up to the current **cmd>** prompt line.

Note

- Clearing the console increases console messaging performance.
- Clearing the console does not clear the contents of the ACE log file.

Viewing the ACE Log File

All Tcl commands and messages issued during an ACE session are recorded in the ACE log file. If the text limit is reached from excessive messages, it is sometimes useful to browse the log file for previous messages. To open the ACE log file in the editor area, simply click the ( ) **Display Log File** toolbar button in the Tcl Console view.

Object Type Prefixes

There are a variety of different object types supported by ACE. Most of these object types have a special single-letter prefix designating the type. These type prefixes are useful to avoid name collisions (i.e., between a net and a pin with the same name).

Many Tcl commands (i.e., `select`) require that these prefixes be used when commands are issued. Other commands (i.e., `find`), by default, include these prefixes on the return values.

Table 144: ***Object Type Prefixes Used in ACE Tcl Commands (Sorted Alphabetically by Prefix)***

Prefix	Object Type
`c:`	Critical Path
`d:`	Device Port
`f:`	Fabric Pin
`i:`	Instance
`k:`	Clock Domain
`n:`	Net
`p:`	Port
`s:`	Site
`t:`	Pin

Table 145: ***Object Type Prefixes Used in ACE Tcl Commands (Sorted Alphabetically by Object Type)***

Object Type	Prefix
Clock Domain	`k:`
Critical Path	`c:`
Device Port	`d:`
Fabric Pin	`f:`
Instance	`i:`
Net	`n:`
Pin	`t:`
Port	`p:`
Site	`s:`

Creating an IP Configuration

Achronix FPGAs feature a wide variety of embedded IP. These highly flexible IP blocks require configuration for proper operation.

ACE includes a number of IP editors (see page 26) and views (see page 31) which work together to provide a guide through the process of correctly configuring IP. The data for these IP configuration editing sessions is stored in `.acxip` files, which may be saved and loaded for future reuse or modification.

Using the data stored in the `.acxip` files, ACE generates RTL wrappers (Verilog and VHDL) containing the specified configuration parameters around the appropriate Achronix macro cells, as well as appropriate `.sdc` and `.pdc` files to complete the IP timing and pre-placement configuration. These generated files may then be incorporated into the user design for synthesis and simulation.

> **Note**
>
> Use of a generated VHDL wrapper also requires the generated Verilog wrapper (the VHDL simply wraps the Verilog instantiation).

Creating and editing IP configurations is typically performed from the ( ) IP configuration perspective (see page 24). In addition to the IP Configuration editors, this perspective incorporates supporting views allowing:

- Creating new IP configurations (IP libraries view (see page 81))
- Viewing a graphical diagram of the IP configuration currently being edited (IP diagram view (see page 80)); the diagram may show the macro interface, the dataflow, and/or the placement of the IP instance within the chip

- Navigating instantly to any page of the active IP configuration editor, while displaying the names and validity of each page (outline view (see page 112))
- Viewing a detailed list of all the errors and warnings pertaining to all IP configuration files currently opened (IP problems view (see page 81))
- Navigating directly to the source of the problem in the relevant IP configuration editor (IP problems view (see page 81))

The major subtasks regarding IP Configuration management are covered in the following sections.

Creating and Naming an IP Configuration

Switch to the IP configuration perspective either by clicking the ( ) IP configuration perspective icon or selecting **Open Perspective → IP Configuration** from the main menu. Select **File → New → IP Configuration...** from the main menu, or use the IP libraries view (see IP libraries view (see page 81)) to open the New IP Configuration Dialog (see page 166). After setting the location and name for the `.acxip` configuration file, click **Finish** to complete the process and activate the appropriate IP editor.

Caution!

The file name chosen for the `.acxip` configuration file is used as the module name for the generated Verilog module. A name must be chosen for the `.acxip` configuration file that is both a valid file name and also a valid Verilog module name which does not conflict with any other module names defined in the Achronix libraries.

For example:

- If the `.acxip` configuration file is named `foo.acxip`, the generated Verilog module is named `module foo`.
- If the `.acxip` configuration file is named `LRAM.acxip`, the generated Verilog module is named `module LRAM`.

If `LRAM` is a primitive in the Achronix libraries, the user design errors out in simulation or synthesis with module name conflicts.

Setting the IP Configuration

From the IP Editor, use either the **« Back** and **Next »** or the Outline view (see page 112) to navigate the editor pages, setting the appropriate values needed for the desired configuration. Any errors and warnings are displayed in the IP Problems view (see page 81). Some IP editors also display supplemental graphical information in the IP Diagram view (see page 80).

In addition to the list of problems within the IP problems view, to the left of most fields (sometimes also called properties) there is a button with an icon indicating the validity of the value in that field. The green checkmark ( ) indicates the value in the field has no problems. A warning ( ) or error ( ) icon is shown when the field value does have one or more problems. The tooltip for the button then shows the problem being reported for the associated field. Clicking the button transfers the application focus to the IP problems View, and within that view, selects all the problems associated with that field.

Note

In complicated IP, where there are many interactions between fields, there might be more than one problem entry associated with a single field.

Editable Fields

Most of the properties/fields within the IP Editor pages are editable and correspond (sometimes loosely) to parameters in the underlying Verilog macros.

Editable fields are meant to be modified in a top-down, left-to-right order. This order is recommended because some of the field values affect the validity of downstream values, and ACE often tries to help keep configurations legal by automatically changing downstream values when they are incompatible with newly-edited upstream values.

Often, editable fields may be temporarily disabled when upstream choices cause the field to become irrelevant, or to have only a single legal value. Disabled editable fields become read-only and are shown with an alternate background color, typically grey.

Tip

Modifications should be made to fields within the IP configurations editors in a top-down, left-to-right order. When editing an upstream value, it often causes downstream values to be overwritten without warning.

Calculated Fields

Some of the fields in the IP editor pages are never editable. These fields contain calculated values based upon the current contents of user-editable fields. These calculated fields are provided for informational purposes.

Many of these calculated values have limited ranges of legal values — when the calculated value falls outside the legal range, the calculated value background color changes to indicate a problem. As when user-editable IP configuration properties fall outside a legal range, an IP problem entry (see IP problems view (see page 81)) is created. But an IP problem created by a calculated value field does not "blame" the calculated value field, it instead blames one of the user-editable properties involved in its calculation. While only one field is blamed in an IP problem entry, be aware that all active fields that might be involved in the calculation are listed in the IP problem entry as potential fields which, when changed, might fix the IP problem.

Note

While only one field is allowed to be blamed in an IP problem entry, remain aware that all active fields that might be involved in the calculation are listed in the IP Problem entry. Any one of these listed fields, when changed, might fix the IP Problem.

IP Editor Navigation

Navigate between sequential IP editor pages by using the **« Back** and **Next »** buttons. When one of these buttons becomes disabled, it means there are no further pages of configuration information in the indicated direction.

The currently active page is always selected in the outline view (see page 112). Navigate directly to a given IP configuration page simply by selecting the desired page name in the **Outline View**. Be aware that pages may be created or removed from the outline view based upon user changes to the IP configuration editable fields.

Left-click any text in the IP diagram view (see page 80) to turn to the IP configuration editor page containing the settings for that text.

Double-click a table entry in the IP problems view (see page 81) to turn to the IP configuration editor page containing the property being blamed for the selected IP problem.

Generating the IP Design Files

After setting the IP configuration, click the ( ) **Generate IP Design Files** icon to open the Generate IP Design Files dialog (see page 163). Select the desired options such as whether to generate the Verilog wrapper, VHDL wrapper, timing constraints, placement constraints, etc. After selecting the desired options and file paths, click **Finish** to create the selected files.

> **Note**
>
> The generated VHDL RTL wrapper is not standalone. It requires the generated Verilog RTL file.

Adding Configuration Files to a Project

Existing configuration files (from other projects, or that were removed from the current project at some point) can be manually added to the active project. Use the procedure under adding source files (see page 272) to add the configuration file and its related source files to the active project.

Live Link Tuning for SerDes and Derived Interfaces

> **Note**
>
> **Live Link Tuning using the ACE GUI is presently only available for Speedster22i FPGA devices.**
> Live Link Tuning is not available for other Achronix product types.

> **Warning!**
>
> **This section is only a summary; external references contain complete details.**
>
> A complete GUI Link Tuning reference is available as a separate document, the *SerDes Link Tuning GUI User Guide*. Despite the specificity of the document name, that reference is presently expected to cover the GUI Link Tuning behavior for all Speedster FPGA SerDes-derived interfaces requiring link tuning.
>
> Other reference documents are planned for alternate (non-GUI) link tuning procedures, including (in some cases) automated tuning that can be managed from within the logic of user designs (without requiring manual user intervention).
>
> The remainder of this section of the *ACE User Guide* should be considered only an overview of the GUI link tuning procedure. See the external reference(s) for more complete link tuning details, including valuable troubleshooting information.

Several of the high-speed interface IP flavors on Achronix Speedster FPGAs are derived from the SerDes interface, or potentially use a specialized subset of the full SerDes functionality. The SerDes PMA Rx Equalization must be tuned specifically to each type of application, board, usage, or data pattern to get optimal link quality. The Rx Eq settings used for a 10G Ethernet application might not work for a 10G Interlaken application, and optimized values for one board may or may not be optimal values for another board with that same application. That said, a group of settings can usually be found that work across a wide range of applications/boards for a given protocol.

The Achronix Speedster SerDes PMA hardware comes equipped with an adaptive equalization engine that enables computing new Rx Equalization settings to find the best possible eye opening for the link. The adaptive equalization feature (Auto Eq) can be run via a sequence of register reads / writes to the SerDes PMA. As a result, new PMA Rx equalization settings are computed and stored in the PMA registers.

The hardware also allows measurement of the quality of the Rx four-point eye opening, which is translated into a single number called the **Figure of Merit** (FOM). The larger the Figure of Merit, the larger the eye opening. The FOM is also captured using a sequence of PMA register reads and writes.

To access the SerDes Link Tuning feature in the ACE GUI, go to the IP Configuration Perspective (see page 24) and open the existing ACXIP file for the SerDes, Interlaken, or Ethernet interface. This opens the appropriate IP Configuration Editor for the chosen IP. While each of these editors includes its own Link Tuning Page, the pages are currently identical for all three IP flavors. The Link Tuning pages support simultaneous tuning of 1 to 12 SerDes lanes (as defined by the Number of Lanes on the Overview page of each ACXIP file). The Rx PMA Eq and Tx PMA Driver parameters may be viewed and altered (for all used lanes) live on these pages, but there is presently no support for automated tuning of the Tx parameters.

> **Note**
>
> **A properly configured ACXIP file is required before beginning Link Tuning.**
>
> It is assumed that customers are using the ACE IP Configuration GUI to generate the wrappers for the SerDes (etc.) interface when integrating the SerDes (etc.) IP into RTL. As a result, an existing ACXIP file must be had before commencing GUI-managed Link Tuning. If an ACXIP file is not available, a new SerDes, Interlaken, or Ethernet ACXIP file can be created (see Creating an IP Configuration (see page 306)) and configured to match the settings used in the design, including (very important) the exact number and placement of the SerDes lanes. A properly configured ACXIP file is an absolutely necessary starting point for the GUI Link Tuning process.

The GUI Link Tuning pages interact with a live FPGA through the Bitporter JTAG interface. It is assumed that the ACE <-> FPGA JTAG connection is already properly configured, since it is also required when Playing a STAPL File (Programming a Device) (see page 350), a necessary precursor to Link Tuning.

> **Note**
>
> **The Bitporter/JTAG connection must be configured before using the Link Tuning functionality!**
>
> During Link Tuning, ACE interacts with the FPGA using the JTAG interface through a Bitporter pod. This Bitporter / JTAG interface must be properly configured in ACE before using the Link Tuning functionality. The configuration is managed using the Configure JTAG Connection Preference Page (see page 186). See Configuring the JTAG Connection (see page 331) for more details.

A Summary of the Link Tuning Process

The Link Tuning page is a portion of the full IP Configuration Editor. As such, the rest of the IP Configuration Editor pages must be properly configured for the interface to be tested, before beginning tuning. Most importantly, the configured number of lanes and the lane placement must be correct, and identical to the SerDes lanes being used on the running FPGA.

1. Create a Speedster SerDes, Interlaken, or Ethernet IP Configuration by creating a new ACXIP file and filling in all needed configuration options. See Creating an IP Configuration. (see page 306)
2. Complete the configuration process for the selected IP. Pay special attention to the proper setting of the number of lanes, and the placement of the lanes on the FPGA. See Setting the IP Configuration (see page 307).
3. Save the ACXIP file and generate the IP design files (RTL, SDC, PDC, etc). Integrate these files into the full design. See Generating the IP Design Files (see page 309).
4. Run the design through Synthesis, and then Place and Route. When timing is met, generate a bitstream. See Running the Flow. (see page 280)
5. Make sure all ref clocks are connected and running on the board, including for each of the SerDes lanes used in the design.

> **Note**
>
> **Do not forget the ref clocks!**
>
> Connecting and running the ref clocks to the utilized SerDes lanes is the detail most often forgotten when attempting Link Tuning. Save everyone a support call, and double-check that the clocks are connected and running to all the lanes configured in the ACXIP file before starting link tuning! There is an easy-to-read Placement Diagram in the IP Diagram View (see page 80) which shows the exact lanes being used by the current ACXIP file. As an example, see the diagram for the Speedster22i SerDes Configuration Editor.

6. Program the FPGA with the generated bitstream (from step 4). See Playing a STAPL File (Programming a Device) (see page 350).
7. Run either real data traffic or an appropriate test pattern (PRBS31, etc) which closely matches the expected data pattern/encoding into the FPGA over the link to be tuned. This is used for the initial FOM capture.
8. Open the ACXIP file (the same one used during the above preparation) in ACE and go to the Link Tuning Page.
9. Press the **Update from Chip** button to see the status of the PMA and initial FOM for each lane.

> **Note**
>
> **Do not proceed to the next step until all the PMA Status lights for all lanes are green!**
>
> After the update from chip completes, if the PMA status is not all green (good) for a given lane, then no FOM can be computed, and Rx Auto Eq cannot be run on that lane. The issue with the PMA status must first be fixed before continuing. The fix might involve manually changing some Tx Driver or Rx Eq settings to get the status to become green (good).

10. When the PMA status is good, observe the initial FOM for each lane. It is good practice to write down the FOM values, along with the Rx and Tx settings used. Taking screenshots is an easy alternative (ACE presently does not log the full tuning process).
11. Press **Run Rx Auto Eq** and wait for it to complete.

> **Warning!**
>
> **Rx Auto Eq tuning should not be run with live traffic over the link.**
>
> While FOM capture may be performed as a background procedure at any time with live data, the Rx Auto Eq tuning algorithm cannot.
>
> Rx Auto Eq tuning is not supported as a background process. Changing any Rx Eq parameter (including those affected during automated tuning) while receiving live traffic results in bit errors. Rx Auto Eq tuning should only be performed as a foreground tuning procedure while sending idle characters.

12. Observe the new FOM for each lane. The new FOM should be much better than the initial FOM. Again, it is good practice to capture the new FOM along with the settings used.
13. Press **Sync GUI with Chip** and then save the ACXIP file (**File -> Save**) to store the new tuned Rx Eq settings back into the source ACXIP file.
14. Re-generate the IP design files (RTL, SDC, PDC, etc) from the ACXIP file and double-check that the newly updated files are included in the full design.
15. Re-run Synthesis, Place and Route, and Bitstream Generation to capture the new optimal Rx Eq settings back into the bitstream.

Viewing the Floorplanner

This section covers working with the floorplanner.

Opening and Closing the Floorplanner's Fly-Out Palette

To open and close the Floorplanner view fly-out palette of view options:

1. Click the (◀) **Fly-out** button on the far right side of the Floorplanner View (see page 53) to open the fly-out palette.

   > **Note**
   >
   > While the fly-out palette is open, it may be resized by clicking and dragging its left border.

2. When the view options are configured, click the (▶) **Fly-in** button on the left side of the fly-out palette to close the fly-out palette.

Zooming the Floorplanner In and Out

There are several ways to zoom in and out in the Floorplanner View (see page 53).

> **Note**
>
> Zoom levels are always in powers of 2 (i.e., zoom in is at 200% and zoom out is at 50%). Therefore, it might not be possible to zoom in to perfectly fit a given area.

To zoom in and out with the mouse wheel:

1. Hover the mouse cursor over the desired point from which to zoom in or out in the Floorplanner view.
2. Move the mouse wheel forward to zoom in or backward to zoom out.

To zoom in and out using keystrokes:

1. Hover the mouse cursor over the center of the desired area from which to zoom in or out in the Floorplanner view.
2. Type either "Z" or "+" on the keyboard to zoom in or "z" or "-" to zoom out.

To zoom in and out using the **Zoom Tool**:

1. Select the ( ) **Zoom Tool** from the view toolbar.
2. To zoom in on an area, click in the upper left corner of the area desired and drag the mouse to the lower right until the zoom rectangle encloses the area desired. To zoom out, click the point on the Floorplanner view from which to zoom out and drag the mouse to the upper left until the zoom out label indicates the desired zoom level.

To zoom in and out with the **Placement Tool**:

1. Select the ( ) **Placement Tool** from the view toolbar.
2. Hover the mouse cursor over the point from which to zoom in or out in the Floorplanner view. Click the left mouse button to zoom in or the right mouse button to zoom out.

To zoom in and out with the **Zoom In** and **Zoom Out** buttons:

1. Pan to the area from which to zoom in to or out in the Floorplanner view.
2. Click the ( ) **Zoom In** button to zoom in or the ( ) **Zoom Out** button to zoom out.

Floorplanner Panning

To pan with the scroll bars:

1. Click and drag the vertical scroll bar to pan up and down or click and drag the horizontal scroll bar to pan left and right.
2. In Linux, place the mouse cursor over a scroll bar, then roll the mouse wheel.

To pan with key-strokes:

1. Use the arrow keys on the keyboard to pan left, right, up and down.
2. To scroll faster, press the CTRL key while pressing the arrow keys.

To pan with the **Placement Tool**:

1. Select the ( ) **Placement Tool** from the view toolbar.
2. Hover the mouse cursor over any point in the Floorplanner view which shows the pan cursor, typically a variant of ( ), though in some OS flavors and themes this pan cursor appearance may vary widely. Click and drag the view with the mouse to pan around.

> **Tip**
>
> If panning the view is too slow, in the Floorplanner View Optimizations Preference Page (see page 195), there is a setting **When panning, show only background layer** to improve panning/scrolling performance by reducing the amount of graphic rendering performed during the pan/scroll operation.

Selecting Floorplanner Objects

To select objects with keystrokes:

1. In the ( ) **Selection** section of the view fly-out palette, check the object types to select.
2. Press and hold the S key on the keyboard to start a selection rectangle at the current mouse cursor position (clearing/replacing the previous selection).
   Optionally, press and hold the SHIFT+S keys to start a selection rectangle at the current mouse cursor position (adding to the current selection instead of replacing it).
3. Drag the mouse while holding down the key or keys on the keyboard to create a selection rectangle which includes the objects desired.
4. Release the key(s) to apply the selection.

To select objects with the Selection Tool:

1. Click the ( ) **Selection Tool** on the view toolbar.
2. From the **Selection** section of the view fly-out palette, check the object types you wish to select.
3. Also, ensure the **Action** control in the fly-out palette is set to **Select**.
4. Click the desired object, or click and drag with the left mouse button in the view to create a selection area rectangle (clearing/replacing any previous selection).

Optionally, hold the CTRL key when clicking/dragging (adding to the previous selection instead of replacing it).

5. Release the mouse button to apply the selection.

Additionally, right-click individual objects and select **Add to Selection** from the context menu popup.

Further details about the ACE selection set are available on the Selection View (see page 136) page.

Deselecting Floorplanner Objects

To deselect objects with key strokes:

1. Select the ( ) **Selection Tool** from the view toolbar.
2. From the **Selection** section fly-out palette, check the object types to deselect.
3. Press and hold the D key on the keyboard to start a selection rectangle at the current mouse position.
4. Drag the mouse while holding down the key to create a selection rectangle including the objects to deselect.
5. Release the D key to remove the objects within the rectangle from the current selection set.

To deselect objects with the **Selection Tool**:

1. Select the ( ) **Selection Tool** from the view toolbar.
2. From the **Selection** section of the fly-out palette, check the object types to deselect. Also, ensure the Action control is set to **Deselect**.
3. Click and drag with the left mouse button in the view to create a selection rectangle.
4. Release the mouse button to remove the objects from the current selection set.

Toggling Floorplanner Mouse Tools

To toggle the mouse tools:

1. Press the ALT key on the keyboard to switch between tools, or simply click the desired mouse tool on the view toolbar.

Filtering the Floorplanner View

It is often useful to filter the floorplanner view (see page 53) graphics to see only objects of interest.

Filtering with Layers

Simple filtering of the view by object type is accomplished with the **Layer** options in the view fly-out palette.

By checking or unchecking the individual layers, all members of a given object type may be shown or hidden. Sites, Instances, Clock Routes, and Non-clock Routes may be manipulated in this way.

Because routes are always painted on top of instances, which themselves are painted on top of sites, it is often necessary to disable the painting of the topmost layers when they obscure the lower layers.

> **Note**
>
> The hiding of individual objects of a given object type layer may be overridden if that object is selected or highlighted, depending upon the current preference settings. See the floorplanner view colors and layers preference page (see page 190) for more information about changing these preferences.

Filtering with Selection

Selection overrides the layer filters. When a layer is turned off, selected objects (those in the current ACE selection set) remain visible. So, for example, to see just the selected instances, turn off the instances and routes layers. The selected instances remain visible, while all other placed instances (and all non-selected nets) are hidden.

To filter with selection in the floorplanner view:

1. Add the desired objects to the current ACE selection set.
2. In the ( ) **Layers** section of the fly-out palette, un-check the object types to hide.

Choosing Floorplanner Object Tooltips

For instant feedback on instance, net, or site names in the floorplanner view (see page 53), a tooltip (hover text) can be enabled. In addition, the contents of the tooltip can be printed to the Tcl Console View (see page 144) for easy copy and paste.

To get object tooltip text:

1. In the ( ) **Tool Tip Text** section of the fly-out palette (see page 56), enable the checkboxes for the object types with data that should be contained in the tool tip text.
2. In the floorplanner view, hover the mouse cursor over objects to display the tool tip text.

> **Tip**
>
> **Capturing Tooltip Content:**
>
> Optionally, press the P key on the keyboard while tooltip text is visible to print the tooltip text to the TCL console view, allowing easy copy and pasting to create TCL commands or scripts.

Viewing Floorplanner Object Labels

A variety of object labels are available when displaying objects in the floorplanner view (see page 53) (see "Fly-Out Palette").

To display object labels in the Floorplanner view:

1. In the ( ) **Labels** section of the fly-out palette, select which object labels to display.
2. Pan and zoom to objects of interest to view the object labels.

> **Note**
>
> Some labels are not painted unless the view is zoomed in far enough to display the full extent of the text.

Highlighting Objects in the Floorplanner View

There is typically a tremendous amount of visualization data available in the floorplanner view (see page 53). Because viewing all of the data simultaneously can be overwhelming, ACE provides tools such as selection and highlighting so that a particular subset of the entire design may be visualized within the floorplanner view. For simple, short-term subset visualizations, this functionality is provided by the ACE selection set as managed in the selection view (see page 136). For longer-term visualizations, or to simultaneously compare and contrast multiple design subsets, ACE provides the `highlight` Tcl command. As with the ACE selection set, applied highlights are visibly displayed on placed/routed objects in the floorplanner view. Highlight colors are also shown in several tabular views such as the netlist browser view (see page 89), where the highlight colors of instances are displayed in their own table column.

> **Note**
>
> **Only instances, nets, and paths may be highlighted.**
> Currently highlights are only supported for individual instance, net, and path object types (remember to use the correct object type prefixes (see page 305) when using the Tcl commands).

Most of the views within the floorplanner perspective provide context-sensitive functionality to manage highlights, through buttons in each supporting view to ( ) **Highlight** chosen objects, ( ) **Un-highlight** chosen objects, or to ( ) **Choose Highlight Color** which is next used from that view (each view tracks an active highlight color independently of the other views, allowing one color for a selection highlight, and an alternate color for a netlist browser highlight). Additionally, some of the views (typically those representing multiple aggregations of objects) include a button to ( ) **Auto-highlight** all objects within that view, with each aggregation automatically using a different color.

> **Caution!**
>
> In tabular views (such as the clock domains view (see page 33)), highlights may be shown and manipulated for aggregations of objects (as with the highlight color of a clock domain row representing the shared highlight color of all instances within that clock domain).
>
> Be aware that if there are multiple highlight colors within the aggregation, then no highlight color is shown for the aggregation row in the table. The aggregation only displays a highlight color in the table if every single object within the aggregation has the exact same highlight color.
>
> Additionally, when a new highlight (or un-highlight) is applied to an aggregation, it affects all individual members of that aggregation. The highlight color of all contained individual objects are overwritten with the new highlight value.

The following Tcl commands are available to manage highlights:

- `highlight`
- `apply_highlights`

> **Note**
>
> There is no specific Tcl command to remove existing highlights. Instead, exclude the `-rgb` flag when calling `highlight`, which effectively applies a non-highlight to the specified object(s).

Selection vs. Highlighting

Despite some similarities, selection and highlighting serve two different purposes in ACE. They are compared and contrasted in the table below.

Table 146: *Selection and Highlighting Differences*

Selection	Highlighting
There is a single ACE Selection Set.	Each object in a design may have its own unique highlight color, or a single highlight color may be applied to multiple objects, even if they are different object types.
The Selection color (a very bright green by default) is managed globally for each object type, through the floorplanner view colors and layers preference page (see page 190).	Each Tcl call to `highlight` an object must specify which color to use for that call. When using the ( ) **Highlight** action from within a view, the ( ) **Choose Highlight Color** for that view is used.

Selection	Highlighting
A selection is very short-term, and is never saved /loaded between sessions.	A highlight is expected to be long-term, and highlight colors on objects are saved in `.acxdb` files when Implementations (see page 215) are saved. As a result, prior highlights are restored when an implementation `.acxdb` file is loaded.
Selection may be applied (through the appropriate view) to aggregations of objects	Highlights may also be applied (through the appropriate view) to aggregations of objects.
Selection membership may be managed through the selection view (see page 136)	There is presently no special view to manage highlights.
The selection color of an object (or aggregation) is only rendered within the floorplanner view.	The highlight color is rendered not only within the floorplanner view, but also (sometimes as individual objects, sometimes in aggregate) as a color tile in most of the tabular supporting views of the floorplanner perspective. When the highlight color is displayed in a tabular view, it is typically shown within its own column of color tile values.

Objects May Be Both Selected and Highlighted Simultaneously

It is possible for objects to be both selected and highlighted at the same time. When this occurs, the exact precedence of the color used to render the object is handled differently depending upon the object type, and which view is being rendered.

When paths are displayed in the floorplanner view, the selection color for a path takes precedence over the highlight color (the selection color for a path is managed on the floorplanner view colors and layers preference page (see page 190)).

When instances are displayed in the floorplanner view, the selection color of an instance takes precedence over every other color (the selection color for an instance within the floorplanner view is managed on the floorplanner view colors and layers preference page (see page 190)).

> **Note**
>
> It is possible to change the relative render priorities of some of the states of an instance on that same floorplanner preference page. The instance states (see page 243) section discusses this in further detail.

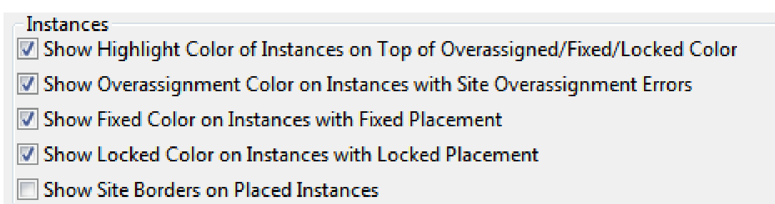


Figure 142: *Floorplanner Preferences Instance State Render Management*

Uniquely among the object types, nets may display both their selection and highlight states simultaneously in the floorplanner view, though this is disabled by default (for performance reasons). By default, when nets are displayed in the floorplanner view, the selection color takes precedence over the highlight color. But nets, being simple lines, are handled specially, as configured in the floorplanner view colors and layers preference page (see page 190). There, it is possible to choose to have the net highlight rendered on top of the net selection, or vice versa. It is also possible to choose which line thickness (width) shall be used to render both the selection line and the highlight line for the net. By making the bottom line thicker than the top line, it is thus possible to have a "halo" effect of one color outlining the other color for the same net(s).

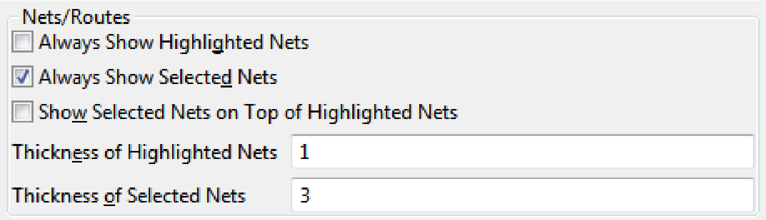


Figure 143: ***Non-Default Preference Configuration Showing Both Selection and Highlights for Nets***

> **Warning**
>
> **Floorplanner Performance:**
>
> Choosing to render highlighted nets thicker than a single pixel wide might have a significant negative performance impact upon floorplanner rendering speeds of large designs at some customer sites. Exact details depend upon specifics of the workstation hardware, OS/kernel version, and the active desktop graphics rendering library.
>
> Rendering selected nets at a thickness greater than one is expected to have little-to-no performance impact upon floorplanner rendering speeds.

Batch Mode Highlighting

Batch-mode highlights in Tcl are supported through the use of the optional `-batch` command-line argument for the `highlight` command. This flag blocks the transmission of incremental highlight updates to the ACE GUI, significantly speeding up execution times when applying multiple scripted highlight changes in sequence. When all batched changes to the highlights have completed, `apply_highlights` must then be called to send the highlight updates to the GUI for display.

```
Example Tcl sequence to highlight all instances orange, except instance names starting with

highlight [find {*} -insts] -rgb {255 128 0} -batch ;# applies orange highlight to all insts
highlight [find {temp*} -insts] -batch              ;# removes highlights from all insts starting with
"temp"
apply_highlights                                    ;# sends pending highlight changes to GUI
```

Pre-Placing a Design

This section details manual pre-placement of instances in ACE.

Placing an Object

Currently in ACE, there are two types of objects that can be placed: instances and ports. Placing a port is equivalent to placing the instance that the port is connected to in the design.

There are two types of manual pre-placement in ACE: soft and fixed. Fixed placement locks the placement of an instance to a site such that the placer is not allowed to move the instance to another site. Fixed placement is the only type of pre-placement command recommended. Soft placement is used as a global placement hint to the placer.

> **Caution!**
>
> Soft placement is not fully functional.

To begin pre-placement activity, the **Run Prepare** flow step for the active implementation must be run. Placing an object automatically resets the flow status to start over from the **Run Prepare** step.

To place an object from the Search View (see page 132), Selection View (see page 136), or Netlist Browser View (see page 89):

1. In the **Placement** ( ) section of the Floorplanner View (see page 53) fly-out palette, check the placement options desired.

    > **Note**
    >
    > Fixed placement is recommended for all pre-placement.

2. Pan and zoom the Floorplanner view until the destination placement site is visible.
3. In the starting view (Search view, Selection view, or Netlist Browser view), click and drag the desired object from the starting view onto the desired destination placement site in the Floorplanner view.

    > **Note**
    >
    > The icon on the now-placed object in the Search view, Selection view, and/or Netlist Browser view is now updated to reflect the placement status.

To re-place an object within the Floorplanner view (move from one placement site to another):

> **Note**
>
> This operation alters an existing site assignment (placement) for an instance. Sometimes it is helpful to start with a fully placed and routed design, and then fine-tune the placement using this operation.
>
> Be aware that changing the placement of an already-placed object clears the current routing data for all connections to and from that instance.

1. In the **Placement** ( ) section of the fly-out palette, check the placement options desired.

    > **Note**
    >
    > Fixed placement is recommended for all pre-placement.

2. Click the **Placement Tool** ( ) on the view toolbar to change the mouse behavior within the view to drag and drop placement mode.
3. Pan and zoom the Floorplanner view until both the to-be-moved instance and its intended destination placement site are visible.
4. In the Floorplanner view, click and drag the placed instance from its current placement site onto the new placement site.

Changing Between Fixed and Soft Placement

There are two types of placement in ACE: soft and fixed. Fixed placement locks the placement of an instance to a site such that the placer is not allowed to move the instance to another site. Fixed placement is the only type of pre-placement command recommended. Soft placement is used as a global placement hint to the placer.

Caution!

Soft placement is not fully functional.

Fixing placement of soft-placement objects

To fix placement with key-strokes:

1. In the **Selection** ( ) section of the Floorplanner View (see page 53) fly-out palette, check the object types having placement that should be fixed.
2. Press and hold the **f** key on the keyboard to start a selection rectangle at the current mouse position.
3. Drag the mouse while holding down the **f** key on the keyboard to create a selection rectangle which includes the objects desired.
4. Release the key to fix the placement of the enclosed objects.

To select objects with the Selection Tool:

1. Select the **Selection Tool** ( ) from the Floorplanner View (see page 53) toolbar.
2. From the **Selection** section of the fly-out palette, check the object types having placement that should be fixed. Also, ensure the **Action** control is set to **Fix Placement**.
3. Click and drag with the left mouse button in the view to create a selection rectangle. Optionally, hold **CTRL** while dragging to add to the selection.
4. Release the mouse button to fix the placement of the activated objects in the selection.

Note

Not using **CTRL** clears the previous selection!

Un-fixing (softening) placement of fixed-placement objects

To un-fix placement with key-strokes:

1. In the **Selection** ( ) section of the Floorplanner View (see page 53) fly-out palette, check the object types with placement that should be un-fixed.
2. Press and hold the **u** key on the keyboard to start a selection rectangle at the current mouse position.
3. Drag the mouse while holding down the **u** key to create a selection rectangle which includes the desired objects.
4. Release the key to un-fix the placement of the enclosed objects.

To select objects with the Selection Tool:

1. Select the **Selection Tool** ( ) from the Floorplanner View (see page 53) toolbar.

2. From the **Selection** section of the fly-out palette, check the object types with placement to be fixed. Also, ensure the **Action** control is set to **Un-fix Placement**.
3. Click and drag with the left mouse button in the view to create a selection rectangle. Optionally, hold **CTRL** down to add to the selection.
4. Release the mouse button to un-fix the placement of the activated objects in the selection.

> **Note**
>
> Not using **CTRL** clears the previous selection!

Group Placement Mode

> **Caution!**
>
> **Advanced Functionality**
>
> Group placement mode is advanced functionality, and has multiple failure cases. Group placement should only be attempted by expert users who understand all the caveats.

During normal drag-and-drop placement operations (when **Group Placement** is disabled), only the placement of a single instance is altered.

When **Group Placement** is enabled, ACE attempts to *shift* the placement of all instances in the current ACE selection set. Group placement cannot be used on instances that are not already placed – attempting to perform group placement on unplaced instances results in failure.

When group placement is attempted, the placement shift is based upon the relative change in placement site coordinates of a single *anchor* instance. The anchor instance is the instance which is dragged-and-dropped. The relative change is calculated based upon the coordinates of the anchor instance initial site and the destination site.

If the starting site coordinates of the anchor instance are at (X=15000, Y=30000) and the destination site coordinates for the anchor instance are at (X=20000, Y=40000), the coordinate shift is (X=+5000, Y=+10000). For each instance in the selection set, this coordinate shift is applied to that instance starting site coordinates; the resulting X and Y values are the coordinates where the destination site is sought for that instance. If no destination site is found at those adjusted coordinates, the entire group placement adjustment is aborted for all instances, and none of the instances are moved. All instances are left untouched in their initial placements, including the dragged-and-dropped anchor instance.

> **Tip!**
>
> Reminder: The anchor instance must already be a member of the ACE selection set when the drag is initiated. All other instances in the selection set must also already be placed before the group placement drag is initiated.
>
> When choosing a drop location for the anchor instance, keep in mind that all other instances in the selection set must have sites at the same relative coordinate offsets from both the anchor instance starting placement and ending placement. If an ending placement site is not found for any instance at the expected coordinate offsets, the entire group placement adjustment operation is aborted, and all instances remain in their initial placements.

Adjusting the Existing Placement of a Group of Selected Instances

1. Empty the ACE Selection Set (as seen in the Selection View (see page 136)).
2. Select all the instances that should take part in the group placement adjustment, so that they are added to the ACE Selection Set.
3. Ensure that all instances in the ACE Selection Set are already placed (the icon for the instances in the Selection View should be either ✔ for Soft Placement or 🔒 for Fixed Placement).

4. Ensure that the **Fixed Placement** checkbox (within the Floorplanner View (see page 53) fly-out palette) is in the desired state.
5. Enable **Group Placement** mode by clicking the checkbox (within the Floorplanner View fly-out palette).
6. Choose which placed, selected instance is to be the anchor instance, then scroll and zoom the Floorplanner until both the initial site and destination site for the anchor instance are plainly visible.
7. Double-check that all other instances in the selection set have corresponding destination sites.
8. Drag the anchor instance from its initial placement to the destination site. If all initial requirements are met, and if destination sites were found for all selected instances at the calculated offsets, the GUI will issue a bulk Tcl `set_placement` command for all selected instances, and the instances should move to the new sites.
9. Disable **Group Placement** mode by clicking the checkbox (within the Floorplanner View fly-out palette).

> **Caution!**
>
> **Group Placement pays attention to the Fixed Placement checkbox:**
>
> Be aware that when a group placement adjustment succeeds, when the new placement is applied, the current state of the Floorplanner View **Fixed Placement** checkbox affects all adjusted placements.
>
> If **Fixed Placement** is checked, all adjusted placements are fixed, regardless of whether they initially had soft or fixed placement. If **Fixed Placement** is unchecked, all adjusted placements are soft, regardless of whether they initially had soft or fixed placement.
>
> All routing to and from the moved instances will be cleared after the placement.

See also: Floorplanner View (see page 53), Selection View (see page 136), Selecting Floorplanner Objects (see page 313), Deselecting Floorplanner Objects (see page 314), `select` , `deselect` , `set_placement`

Removing Placement

To un-place objects with key-strokes in the Floorplanner View (see page 53):

1. Ensure the **Instances** checkbox is checked in the **Selection** ( ) section of the view Fly-out Palette.
2. With the mouse positioned in the Floorplanner view, press and hold the **r** key on the keyboard to start a selection rectangle at the current mouse position to remove placement of objects.
3. Drag the mouse while still holding down the **r** key to create a selection rectangle including the objects to be un-placed.
4. Release the key to un-place all the objects contained by the selection rectangle.

To un-place objects with the **Selection Tool** ( ) in the Floorplanner view:

1. Select the **Selection Tool** ( ) from the view toolbar.
2. In the **Selection** ( ) section of the view fly-out palette:
    a. Set the Action control to **Remove Placement**.
    b. Ensure the **Instances** checkbox is checked.
3. Click and drag with the left mouse button in the view to create a selection rectangle.
4. Release the mouse button to un-place all the objects within the selection rectangle.

It is also possible to un-place objects using the right-click context menu in the Floorplanner view, Search View (see page 132), and Selection View (see page 136).

The fastest way to un-place multiple objects is to add them all to the ACE Selection Set (as shown in the Selection view; see Selecting Floorplanner Objects (see page 313)), and then un-place all of them at once by performing any one of the following:

- In the Selection view, right-click the mouse on the **Instances** node in the tree, then choose **Unplace All Instances in ACE Selection Set**.
- In the Floorplanner view, right-click the mouse anywhere on the floorplan, then choose **Unplace All Selected Instances**.
- In the Tcl Console View (see page 144), type: " `run_unplace -insts [get_selection]` ". (See `run_unplace`.)

Performance Tip:

It is always faster to un-place multiple objects at once instead of individually, especially when a very complex net (like a clock net) is affected.

Saving Pre-Placement Constraints

To save the current placement to disk as pre-placement constraints in `.pdc` files:

1. Place objects in the design as described in Placing an Object (see page 318).
2. In the Floorplanner view or Package view, click the **Save Pre-placement Constraints** toolbar button () to bring up the Save Placement Dialog (see page 170).
3. Configure the dialog with appropriate options and click **Finish**.

After clicking **Finish**, the pre-placement files are saved to disk. If the option was selected to automatically add the files to the current project, the Projects view shows the new files under the active project Constraints folder.

Using Pre-Placement in the Flow

Types of Pre-Placement

There are three ways to pre-place instances in ACE:

- After the **Run Prepare** flow step, interactively pre-place instances using the ACE GUI drag-and-drop placement features, or use the `set_placement -fixed` TCL command in the TCL Console
- Include a PDC constraints file in your ACE project that uses `set_placement -fixed` TCL commands to pre-place the instances
- Set the location parameter on the instance primitive in the user design RTL (not recommended; the location parameter is effectively the same as using the `set_placement –fixed` Tcl command on that instance)

Using the `set_placement -fixed` Tcl commands in a PDC constraints file is recommended. If using the location parameter in the user design RTL, the RTL must be changed and re-synthesized to update the pre-placement.

ACE applies pre-placement from user design RTL and PDC constraints at the end of **Run Prepare** in two stages:

1. ACE loops over all instances that have the location parameter set and internally calls `set_placement –fixed` on each instance and then prints the log file message “Applying defparam placement of <instance> to location <location>”. The log file or TCL console shows messages for each instance placed with the location parameter.

2. The **Run Prepare** step applies all of the `set_placement` commands in the PDC files. If there is a `set_placement` command in your PDC for an instance that is already placed with the location parameter, the PDC `set_placement` overrides the placement set in the location parameter. There is no warning message. It is the same as having two `set_placement` commands in the same PDC file that place the same instance. The last `set_placement` command always wins.

It is recommended to use only the PDC method.

Recommended Typical Flow

To use pre-placement in the flow, it is recommended to first create the pre-placement constraints:

1. Run the **Run Prepare** flow step on the active implementation.
2. Switch to the Floorplanner perspective and place all the objects for pre-placement (using fixed placement). See Placing an Object (see page 318) for details.
3. Save the pre-placement and automatically add it to the project (see Saving Pre-Placement Constraints (see page 323)).
4. Optionally, a pin assignment report can be generated for the current placement with the `report_pins` (see page 580)Tcl command.
5. Resume running the flow. The pre-placement data is used in the place and route solution.

When the pre-placement constraints are in place, it is recommended to include them in the project for future runs:

1. The next time the flow is run with this implementation, ensure that in the Options View (see page 103), the new pre-placement constraints files are enabled.
2. Simply run the **Run Prepare** flow step. The pre-placement constraints are automatically applied. A pin assignment report is also automatically generated during **Run Prepare**.
3. Optionally, to see that the objects are pre-placed, switch to the Floorplanner perspective to view the placement.

Analyzing Critical Paths

Critical paths are computed by timing analysis. Timing analysis can be run at several points in the flow (see page 221), as indicated in the flow view (see page 61). Timing analysis can be repeated with different implementation (see page 215) options without having to re-run the rest of the flow, by double-clicking the appropriate **Run Timing Analysis** flow step (see page 221).

The results of timing analysis are shown in a timing report (see page 227), which is automatically displayed as timing analysis completes. The most recently generated version of each timing report file is always available in every implementation `reports` sub-directory.

The active critical paths may also be viewed in the critical paths view (see page 48) and the critical path diagram view (see page 45). Unlike the reports, the views only show the critical paths for the active project and implementation (see page 221). When the active implementation changes, the two views are cleared. Also, the views are only populated when timing analysis is run for the active implementation during that same ACE session. The timing analysis data is not saved in the `.acxdb` file, and must be re-created every session to guarantee correctness.

> **Tip**
>
> While a generated timing report may be viewed from an implementation `reports` directory at any time, including in later ACE sessions, the two critical path views only show data from the most recent timing analysis within the current ACE session.

Generating Timing Reports

A timing report (see page 227) is generated and displayed in the GUI whenever one of the **Run … Timing Analysis** flow steps (see page 221) is run. Timing reports may also be generated at any time from Tcl by running the appropriate flow step (`run -step <flow_step_name>`) or with the `run_timing_analysis` Tcl command.

Timing reports can be found in the implementation (see page 215) reports directory, available for browsing via the projects view (see page 125). In addition to the HTML report files displayed in the GUI, there are equivalent report files in text and `.csv` (spreadsheet) formats.

The Timing Analysis implementation options (see page 215) in the options view (see page 103) determine how timing analysis is run and the amount of report information which is generated.

Critical paths are also displayed in the critical paths view (see page 48). The path ID can be used to cross-reference between the critical paths view and the timing reports.

ACE also allows generating a timing report across all temperature corners. This is the default behavior for a complete flow (i.e., `post_process` is complete).

ACE supports place and route across all temperature corners (i.e., place and Route is optimized and timing reported across all temperature corners by checking **Enable all-corner PnR optimization** from the **Options** → **Timing Analysis** menu item (please refer to the table: Timing Analysis Implementation Options under Options View (see page 103)).

The corresponding example ACE Tcl command (see page 534) to set the required implementation option to enable this feature is:

```
set_impl_option -project "wgl" -impl "impl_1"  "pnr_optimize_corners"  "1"
```

Highlighting Critical Paths

> **Note**
>
> The Floorplanner can only display routed paths. Paths which are not routed cannot be displayed in the Floorplanner.

To highlight a routed critical path in the Floorplanner View (see page 53):

1. First, run one of the timing analysis flow steps to generate critical path data.
2. Then, in the Critical Paths View (see page 48), browse through all reported critical paths.
   a. By default, highlight colors of setup/hold violations are arranged in a gradient from red to yellow according to the slack's distance from zero.
   b. Paths with a positive slack (setup/hold met) are colored green by default.
3. To highlight a path in the Floorplanner, simply check the box for the desired path in the Highlight column of the table within the Critical Paths View (see page 48). To un-highlight a path, simply uncheck the box.

> **Tip: Critical Path Highlight Colors May Be Changed**
>
> The highlight color of each individual critical path can be changed by clicking on the color chooser box in the Highlight column of the Critical Paths View (see page 48) table. In the color chooser dialog, select the desired color for that path and click **OK**.

Selecting Critical Path Objects

In order to manipulate objects (for example, by pre-placing them) on a critical path, it is convenient to add them to the current ACE selection set (as displayed in the selection view (see page 136)) for easy access.

> **Note**
>
> When objects are in the ACE selection set, they change to the selection color which overrides all other colors, including highlight colors.

To add a critical path to the current selection:

1. In the critical paths view (see page 48), click the table row containing the data for the path for which objects are to be selected.
2. To add the path to the current ACE selection set, click the ( ) **Select Path** toolbar button on the critical paths view (see page 48) toolbar.
3. The path is now added to the selection in the selection view (see page 136) and is shown with the selection color in the floorplanner view (see page 53).

To add a the pins of a critical path to the current selection:

1. In the critical paths view (see page 48), click the table row containing the data for the path for which objects are to be selected.
2. To add the path pins to the current ACE selection set, click the ( ) **Select Pins** toolbar button on the critical paths view (see page 48) toolbar.
3. The pins are now added to the selection in the selection view (see page 136) and are shown with the selection color in the floorplanner view (see page 53).

To add the instances of a critical path to the current selection:

1. In the critical paths view (see page 48), click the table row containing the data for the path for which objects are to be selected.
2. To add the path instances to the current ACE selection set, click the ( ) **Select Instances** toolbar button on the critical paths view (see page 48) toolbar.
3. The instances are now added to the selection in the selection view (see page 136) and are shown with the selection color in the floorplanner view (see page 53).

To add the nets of a critical path to the current selection:

1. In the critical paths view (see page 48), click the table row containing the data for the path for which objects are to be selected.
2. To add the path nets to the current ACE selection set, click the ( ) **Select Nets** toolbar button on the critical paths view (see page 48) toolbar.
3. The nets are now added to the selection in the selection view (see page 136) and are shown with the selection color in the floorplanner view (see page 53).

Zooming to Critical Paths

To zoom the floorplanner view (see page 53) to the region of a critical path:

1. In the critical paths view (see page 48), click the table row containing the desired critical path data.
2. To zoom to the path in the floorplanner view (see page 53), click the ( ) **Zoom to Path** toolbar button on the critical paths view (see page 48) toolbar.

> **Note**
>
> This action only applies to routed designs.

Printing Critical Path Details

To print the details of a critical path to the Tcl console view (see page 144):

1. In the critical paths view (see page 48), click the table row containing the data for the path for which details are to be printed.
2. To print the details text, click the ( ) **Print Path Details** toolbar button on the critical paths view (see page 48) toolbar.

> **Tip**
>
> Critical path details are also available in the timing report (see page 227)

Using Critical Path Diagrams

The Critical Path Diagram View (see page 45) provides a graphical representation of a single critical path. These paths are each selected from the table in the Critical Paths View (see page 48). The graphical representations consist of circular nodes (representing instances) connected by arrows (representing one or more nets).

> **Tip**
>
> To quickly look at the diagrams for all the critical paths:
>
> 1. Make sure both the Critical Paths View (see page 48) and Critical Path Diagram View (see page 45) are visible.
> 2. Click a row in the Critical Paths view table.
> 3. Use the keyboard up arrow and down arrow keys to change which row is selected in the table.
> 4. The Critical Path view diagram is updated to graph the relevant critical path.

Graph Elements

The graphical diagram is made up of nodes and arrows. The information represented by the nodes, arrows, and their supporting text, can vary depending upon the current settings in the diagram fly-out palette (see page 47).

Nodes

The larger circles in the diagram are the primary graph nodes which represent the key instances or turn points on the critical path. Intermediate nodes, when enabled, are smaller circles, representing instances the data passes through while flowing between turn points. Several useful pieces of information are available for each graph node. These may be enabled and disabled via the fly-out palette.

> **Note**
>
> Some information is hidden when the graph node circle is too small to contain it. To see all enabled information, the diagram must be zoomed in. Configurable tooltips can be used to see information that would otherwise be hidden due to insufficient drawing area.

Arrows

The arrows connecting the graph nodes in the diagram represent the nets connecting the object instances and can also display various pieces of information that may be enabled and disabled via the fly-out palette. In addition to the direction of the arrow, the line types making up the arrow also represent important information:

- Bold arrows, visible when the **Intermediate Nodes** fly-out palette setting is disabled, represent one or more nets and any hidden intermediate nodes, lumped into a single abstraction. Bold arrows, since they potentially represent multiple nets and hidden intermediate instances, may only display text for the cumulative time in picoseconds of their **Delays**. **Net Names** and **Fanouts** are never displayed for bold arrows, since they make no sense in this context.
- Thinner arrows are shown when the **Intermediate Nodes** fly-out palette setting is enabled. Each of these represent an individual net. Because these thinner arrows each represent individual nets connecting the instances, the individual net **Fanouts** and **Net Names** may also be displayed for each arrow, in addition to the **Delays**.

Critical Path Diagram Types

Different types of critical paths may have different visual representations. The **Type** column in the Critical Paths View (see page 48) table provides the critical path type of each row.

All path types are displayed as a straight line of objects connected by arrows.

Adding Portions of the Graph to the ACE Selection Set

When the graph has helped track down which nets and/or instances to adjust for timing purposes, it can be helpful to find those objects in the Floorplanner View (see page 53). To do so, use the techniques described in Selecting Critical Path Objects (see page 326), or add specific nodes and arrows from the graph to the ACE selection set, and use the **Zoom to Selection** button in the Selection View (see page 136) to cause the Floorplanner View (see page 53) to scroll and zoom so that all the selected objects are visible.

Viewing Critical Paths in the Schematic Viewer

Currently, ACE does not have its own built-in schematic viewer. Viewing critical paths requires the synthesis tool. In order to facilitate this, ACE can optionally generate a Tcl script with find commands for objects along each critical path.

To view critical paths in the synthesis tool:

1. In the Critical Paths view (see page 48), click the ( ) **Save Script File** toolbar button to open the Save Script File dialog (see page 174).
2. Enter a valid file name in the Save Script File dialog (see page 174) and click **Save**.
3. Source the saved script file from within the synthesis tool Tcl prompt.
4. After the file is sourced, open the synthesis tool schematic viewer.
5. At the Tcl prompt, enter: `select_one_path <path_id>` to view a given path, where the `<path_id>` is the value from the **Path** column of the table in the Critical Paths view, and/or the **Path Id** value from one of the Timing Reports (see page 227).

Applying and Checking Properties

Applying Properties

There are three methods to apply properties, detailed below. More information and examples can be found in Synthesis Optimizations in the *Synthesis User Guide* (UG071).

defparam

Properties can be applied as defparams on a module or module port in the RTL black-box library. Applying defparams is an internal only method and sets the default value of the property for all instances of that module, or instance pins of a port.

RTL Attribute

Attributes can be set in the RTL to show the design intent and to guide both Synplify and ACE. Synplify attributes are detailed in the Synplify help manual. ACE attributes can also be set in the RTL, and these are passed by Synplify to ACE. Attributes can be applied in both Verilog-2001 and SystemVerilog formats, as shown below:

Equivalent methods for applying properties

```
                                   reg my_reg /* synthesis syn_preserve=1, must_keep=1 */;
(* must_keep=1 *)                  reg my_reg /* synthesis syn_preserve=1 */;
(* must_keep=1, syn_preserve=1  *) reg my_reg;
```

In certain cases it is necessary to set both a Synplify and an ACE attribute when if, without the Synplify attribute, the object may be optimized or reduced. For example, if it is required to keep a register, then Synplify will require the `syn_preserve` attribute to ensure the register is in the netlist output to ACE, and ACE will require the `must_keep` attribute to keep the subsequent register. Similar situations arise with directing and controlling fanout, where both `syn_maxfan` (Synplify) and `fanout_limit` (ACE) may be required. This requirement is a result of the fact that Synplify Pro does not propagate its own `syn_*` attributes on to ACE in the gate-level netlist.

set_property Tcl Command

The set_property command can be applied in a PDC file to an object. See set_property (see page 605) for full details.

```
set_property IOSTANDARD {"LVCMOS18"}  {p:led_anode[0]}
```

Checking Whether Properties Were Applied

It is recommended to use the Synplify technology viewer to verify that properties were applied during Synplify. Selecting the object and then selecting "properties" will list the properties which will be passed to ACE. In addition, the netlist can be searched for the appropriate object, and the properties checked.

Within ACE, examine the object using the Properties View (see page 128), or use the display_properties (see page 546), get_properties (see page 565), or get_property (see page 565) Tcl commands.

Configuring External Connections to Hardware

ACE includes features supporting interactions with running hardware through both the JTAG and DCC interfaces.

The ACE JTAG connection utilizes a Bitporter pod or FTDI FT2232H device and (internally) the `acx_stapl_player` to interact with an Achronix FPGA. The JTAG interface is used by the:

- Download view (see page 51)
- Snapshot Debugger view (see page 139)
- HW Demo view (see page 64)

The automated SerDes link tuning also uses JTAG.

> **Note**
>
> For more details on managing the physical connection between the workstation, the Bitporter pod or FTDI FT2232H device, and the FPGA board, see the *JTAG Configuration User Guide* (UG004), as well as any documentation specific to the development board.

The ACE DCC connection utilizes a direct serial connection (over a dedicated USB cable, kept separate from the Bitporter) to interact with the Achronix development board. The DCC interface is used by the HW demo view (see page 64).

Configuring the DCC Connection

The ACE DCC (Demo Command and Control) connection allows the ACE software to communicate with demo and/or reference designs running on Achronix hardware. The DCC connection is managed using the Configure DCC Connection Preference Page (see page 185).

Background Info

The ACE DCC connection utilizes a direct serial RS232 connection through a simple on-board 2-pin interface, over a dedicated FTDI USB Serial Port cable. The DCC interface does not make use of the Bitporter. The DCC protocol is currently not a published standard, and is only meant to be used with Achronix demo/reference designs running on Achronix FPGAs on Achronix boards. The DCC interactions perform individual register reads and writes, and are executed using a handful of simple Tcl commands.

Installing DCC USB Drivers

The necessary USB drivers for the DCC cable are included in the ACE download. They will need to be installed before the DCC cable is connected.

Windows

When running the ACE installer, simply make sure the setting **FTDI CDM USB drivers for the Development Board DCC interface** is checked. When the installer completes, the DCC cable may be connected.

Linux (not currently supported)

The necessary FTDI USB drivers are already included in supported Linux distros. If the distro-supplied USB drivers do not work correctly, the necessary drivers may be downloaded from the FTDI website. Contact Achronix support if you need help finding the Linux drivers.

Configuring the USB drivers

The DCC cable USB drivers automatically choose a serial port to be redirected to the USB cable. It may be necessary to view and/or change which serial port is chosen.

Windows

To see which serial (COM) port is being used by the driver, go to the Windows Device Manager, and look under **Ports (COM and LPT)**. The correct COM port is shown as **USB Serial Port (COM*)**, with the actual COM port number included.

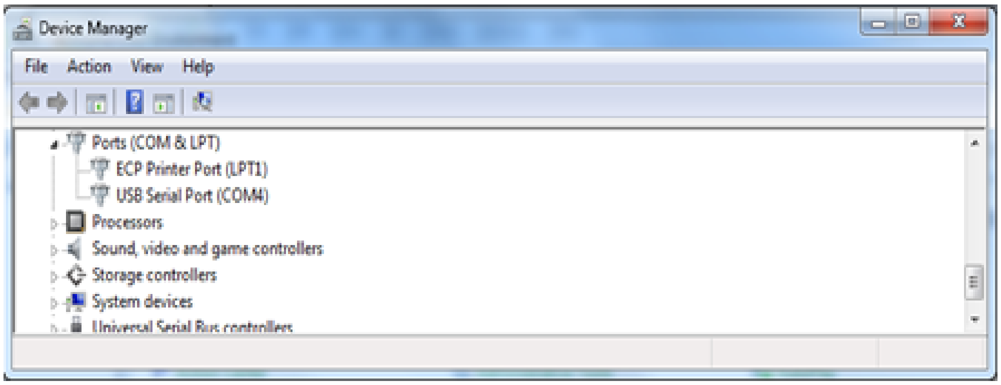


Figure 144: ***Windows Device Manager COM Ports Example***

To change which COM port is being used by the USB cable, right-click the USB Serial Port entry, and then choose **Port Settings** → **Advanced**, and select the alternate COM port to be used.

Linux (not currently supported)

Contact Achronix support for the latest information about finding or changing the DCC cable serial port assignment.

Configuring ACE

When it is determined which serial port is being used by the DCC cable, ACE needs to be configured to use that same serial port.

Open the Configure DCC Connection Preference Page (see page 185) (**Window** → **Preferences** → **Configure DCC Connection**), then populate the **Port Name** field with the serial port name the DCC cable is using. The exact port names used vary according to the operating system in use.

In Windows, this is a COM port, typically one of `COM1` through `COM9`. `COM3` is the most frequent (and thus default) choice. In Linux, the serial ports are named `/dev/ttyS*` with the `*` being replaced by a number. This is typically one of `/dev/ttyS0` through `/dev/ttyS9`.

Configuring the JTAG Connection

Achronix currently supports two JTAG programmer device types: the Bitporter pod, and the FTDI FT2232H device. While in the simplest cases, a test bench may only have a single JTAG programmer device connected to a JTAG scan chain containing a single Achronix FPGA, many may have multiple JTAG programmer devices, and/or multiple devices within the connected JTAG scan chain. For these reasons, the ACE tools must specify which JTAG programmer device connection to use, and which JTAG scan chain member is relevant.

The various tools within ACE obtain their choice of JTAG programmer device and JTAG scan chain configuration from a few different locations, mostly based upon the usage model of the ACE tool itself.

The primary sources of JTAG programmer device and JTAG configuration information include:

- The GUI Configure JTAG Connection preference page (see page 186)
- The implementation options (see page 215), as managed within the options view (see page 103) (the implementation options for the JTAG scan chain values are saved in the STAPL `*.jam` bitstream file during the **Run Bitstream Generation**flow step (see page 221), when that file is being created).
- Command-line overrides

The GUI views responsible for live FPGA interactions retrieve their JTAG programmer device and JTAG connection details from the Configure JTAG Connection preference page (see page 186). Tools that generate a new bitstream for an implementation (typically during the flow (see page 221), but some may also be triggered ad-hoc) retrieve the JTAG scan chain values from the appropriate implementation options (see page 215) as managed by the options view (see page 103) in the **Bitstream Generation** group; these JTAG scan chain values are then saved in the STAPL `*.jam` bitstream file for later use during downloads. Depending upon how the tool is typically used, the various STAPL/download tools may take their settings from:

- The preferences
- The current implementation options
- The STAPL `*.jam` file
- The command line

Note

The `acx_stapl_player` is shipped as a part of the ACE software. While it is used automatically behind-the-scenes by ACE for all JTAG interactions, it may also be used manually from the operating system shell /command-line. The use of this tool is covered in detail within the *JTAG Configuration User Guide* (UG004) (see https://www.achronix.com/documentation/jtag-configuration-user-guide-ug004).

Caution!

Special note for sites with more than one connected FPGA:

The interactive members of the ACE tools currently assume that only a single FPGA is of interest at a time, and those interactive tools all share a single configuration on the Configure JTAG Connection preference page (see page 186). This single configuration is stored per-user, and is not unique per-design or per-implementation.

At sites with more than a single connected FPGA, remember to change the JTAG programmer device and JTAG scan chain values stored on that preference page every time when alternating between FPGAs.

JTAG Programmer Device Connection

It is often possible that multiple JTAG programmer devices are visible from a single workstation. Because of this, ACE software must be configured to know which JTAG programmer device is intended to be used. The GUI JTAG programmer device and JTAG connection details are managed primarily through the Configure JTAG Connection preference page (see page 186) (for interactive GUI tools) and the implementation options within the options view (see page 103) (for tools called within the flow (see page 221)).

Warning!

Bitporter pods may be damaged if improperly connected or power-cycled!

Read the *JTAG Configuration User Guide* (UG004), as well the user guide(s) specific to your development kit, before physically connecting the Bitporter JTAG ribbon cable to the JTAG header on the development board.

Important!

This section describes how to configure ACE to communicate with an already-connected JTAG programmer device. For more details on managing the physical connection between the workstation, the JTAG programmer device, and the FPGA board, see the *JTAG Configuration User Guide* (UG004), as well the user guide(s) specific to the development kit, and any related release notes.

The *JTAG Configuration User Guide* (UG004) also covers additional details about testing JTAG programmer device connections, JTAG programmer device naming/addressing, and managing connections to multiple devices.

ACE uses the `acx_stapl_player` command-line tool to run STAPL programs for all ACE JTAG operations. The `acx_stapl_player` automatically detects the presence of JTAG programmer devices over USB, and Bitporters over Ethernet (if the Bitporter pod is on the same subnet as the workstation running `acx_stapl_player`). If multiple JTAG programmer devices are detected, and ACE has not been informed which of the JTAG programmer devices (by name) should be used, the STAPL program execution fails, because ACE does not pick a JTAG device randomly.

It is strongly recommended to specify the chosen JTAG programmer device by name in all cases, instead of relying upon auto-detection. Not only does this avoid problems if additional JTAG programmer devices are connected at a later date, but connections to named devices are faster to initiate.

Tip

Several seconds of initialization time can be saved on every JTAG connection if the JTAG programmer device is specified by name instead of using auto-detection.

The details of JTAG device naming are covered thoroughly in the *JTAG Configuration User Guide* (UG004). As a simple summary, FT2232H devices are named by their serial number, and Bitporter pods are named with a three-letter prefix for the connection type, and a suffix. Bitporter Pods connected over USB use the prefix `usb` and are addressed by their serial number (which should be visible on a sticker on the pod itself), i.e., `usb12345` for a serial number of 12345. Bitporter pods connected over Ethernet use the `net` prefix and their IP address as the suffix, i.e., `net192.168.1.123` for an IP address of 192.168.1.123.

JTAG Scan Chain

The JTAG (see https://en.wikipedia.org/wiki/JTAG) specification supports multiple devices being connected in sequence, sharing a single set of JTAG pins on the board. These devices are said to be on the same JTAG scan chain. In order for ACE to successfully communicate with any target device in a scan chain, ACE must be told the scan chain configuration on the Configure JTAG Connection preference page (see page 186).

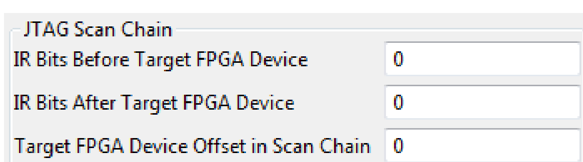


Figure 145: ***JTAG Scan Chain Fields, Configure JTAG Connection Preference Page***

Note

The default JTAG scan chain preference values, all zeros, is always correct for single-device scan chains. For multi-device scan chains, the default values of all zeros never work.

When multiple FPGA devices are attached to the same JTAG scan chain, the target FPGA device must be specified. Because different FPGA devices have different instruction sizes, the total instruction length before and after the target must be specified as well.

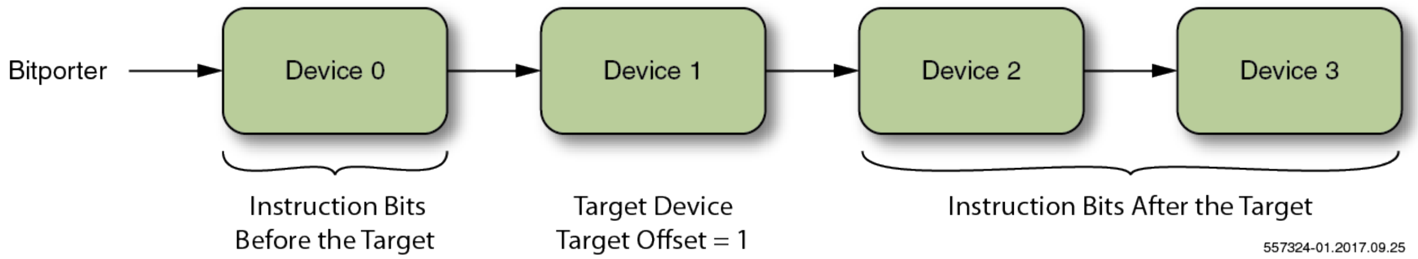


Figure 146: ***Multi-device Scan Chain with Bitporter Example***

The **Target FPGA Device Offset in Scan Chain** specifies the ordinal position relative to the Bitporter. The device closest to the Bitporter (technically, the device closest to the board JTAG TDI pin) has a target offset of 0.

The number of instruction register (IR) bits before the target FPGA device is specified under **IR Bits Before Target FPGA Device**, while **IR Bits After Target FPGA Device** specifies the number of IR bits that follow the target FPGA device in the chain. Achronix FPGA devices have an instruction size of 23 bits. Hence, in the above example, if all devices were Achronix FPGA devices, there would be 23 instruction bits before the target, (23 instruction bits in the target), and 46 instruction bits after the target.

In JTAG, the least significant bit enters the scan chain first, while the most significant bit enters the scan chain last. From the perspective of ACE, *before* refers to the more significant bits in the scan chain, and *after* refers to the less significant bits in the scan chain. Instruction bits before the target are not scanned through the target FPGA. Instruction bits after the target instruction bits are scanned through the target FPGA before arriving at their scan chain destination. The key detail is that ACE regards the before/after terminology from the perspective of where the bits ultimately land in the instruction registers, NOT in terms of when the bits pass through the JTAG TDI pin, and NOT in terms of the sequence in which the bits pass through the target device.

Thus, in a chain of four Achronix FPGAs (each FPGA instruction register consists of 23 bits, the total IR bits = 4 × 23 = 92 IR bits), to specify the device closest to the TDI pin, the initial device (IR scan chain bits [91:69]) requires an entry of 0:69:0 for the three ACE scan chain configuration values. The first 0 for **IR Bits Before Target FPGA Device**, 69 for **IR Bits After Target FPGA Device**, and 0 for **Target FPGA Device Offset in Scan Chain**. The second Achronix FPGA in a chain of four would be 23:46:1, the third would be 46:23:2, and the fourth would be 69:0:3.

Table 147: ***Example Scan Chain Values: Four Achronix FPGAs in the Same Scan Chain***

Device (IR bit Range Within 92-bit IR Scan Chain)	IR Bits Before Target FPGA Device	IR Bits After Target FPGA Device	Target FPGA Device Offset in Scan Chain
0 (bits [91:69])	0	69	0
1 (bits [68:46])	23	46	1
2 (bits [45:23])	46	23	2
3 (bits [22:0])	69	0	3

Specifying a single-device chain (where there is nothing in the chain except the solo target device) would always require an entry of 0:0:0. There are zero IR bits before the target device, zero IR bits after the target device, and it is the device closest to the TDI pin in the JTAG scan chain. This single-device scan chain configuration is the default configuration.

Note

Instruction register lengths vary by vendor and device.

For those new to JTAG, it might be worth mentioning that if non-Achronix devices are in the scan chain, it is extremely likely that their instruction registers are not 23 bits long, thus the before/after bit counts required would not be multiples of 23.

The scan chain Offset number is independent of the IR bit numbers, and is used to derive data register pre- and post-padding, since according to the JTAG specifications, devices being bypassed each always have a DR length of one.

Warning!

Warning for engineers that hand-edit Achronix STAPL (not recommended):

The STAPL programming language (as used in the bitstream `*.jam` files) has an inverse understanding of bits *before* and bits *after* the target device. (STAPL considers *pre* to be the first bits to enter the board TDI pin, and *post* to be the last bits to enter the board TDI pin.) The following table might help clarify the relationships:

Table 148: ***STAPL vs. ACE Terminology Differences: Before/Pre and After/Post***

Description	GUI Phrasing	STAPL equivalent
IR bit count between board JTAG TDI pin and target FPGA device.	IR Bits Before Target FPGA Device	POSTIR
IR bit count between board JTAG TDO pin and target FPGA device.	IR Bits After Target FPGA Device	PREIR
Device count (in scan chain) between board JTAG TDI pin and target FPGA device.	Target FPGA Device Offset in Scan Chain	POSTDR
Device count (in scan chain) between board JTAG TDO pin and target FPGA device.	–	PREDR

Running the Snapshot Debugger

The following sections describe how to configure and use the snapshot debugger in an end-user design.

Note

Snapshot hardware architecture details and use of the `ACX_SNAPSHOT` user macro can be found in the snapshot user guide appropriate to each Achronix device family. These are available at http://www.achronix.com/documentation.html, at the Achronix support site (account required), or from an Achronix FAE.

The Snapshot content here in the ACE User Guide provides a general overview of functions which are common to all Achronix devices, and is focused on the snapshot debugger user interface for real time in-system debugging.

Snapshot Design Flow

> **Warning!**
>
> **The JTAG connection must be configured before using the snapshot debugger.**
> ACE interacts with the FPGA using the JTAG interface through a Bitporter2 pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the Snapshot Debugger view. The configuration is managed using the Configure JTAG Connection preference page (see page 186), which is easily accessible by clicking the ( ) **Configure JTAG Interface** button in the Snapshot Debugger view. See Configuring the JTAG Connection (see page 331) for more details.

Snapshot is the real-time design debugging tool for Achronix FPGAs. Snapshot, which is embedded in the ACE software, delivers a practical platform to evaluate the signals of a user design in real-time and optionally send stimuli to the user design.

To utilize the snapshot debugger tool, the snapshot macro must be instantiated inside the RTL for the design under test (DUT). After instantiating the macro and programming the device, the design can be debugged in the ACE GUI using the Snapshot Debugger view (see page 139) and the VCD Waveform Editor (see page 28), found within the Programming and Debug perspective (see page 24).

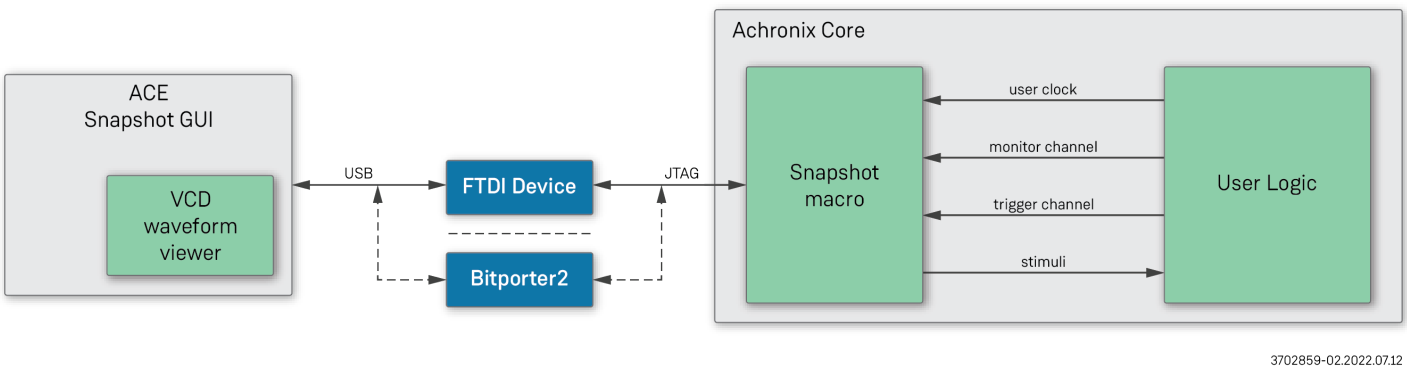


Figure 147: *Snapshot Communication with the Snapshot Debugger View within ACE When instantiated in a design, the Snapshot macro can be used to interface with any logic mapped to the Achronix FPGA core. The Snapshot macro provides a JTAG/JTAP interface to control/observe debug logic mapped to the core. This interface allows the ACE Snapshot Debugger view, which drives the JTAG interface, to control/observe the signals associated with the debug logic.*

Within the ACE GUI, the Snapshot Debugger view allows configuring an embedded Snapshot Debugger core, interactively arm the core, and generate a VCD waveform output of the collected samples. By default, the generated VCD waveform output is displayed in the ACE editor area using the VCD Waveform Editor (see page 28). The VCD output can also be read into a third-party waveform viewer.

At a high level, to utilize Snapshot, first:

1. Instantiate the Snapshot macro `ACX_SNAPSHOT` in the user design.
2. Set the required constraints in the `.sdc` files.
3. Synthesize the design.
4. Place and route the design in ACE.
5. Generate the Bitstream for the design in ACE.

6. Configure the ACE JTAG connection to the FPGA (see Configuring the JTAG Connection (see page 331))
7. Program the Achronix device with the Bitstream.
    - Use of the ACE GUI Download view (see page 51) is documented in the section Playing a STAPL File (Programming a Device) (see page 350)
    - Use of the `acx_stapl_player` executable on the command-line is documented in the *JTAG Configuration User Guide* (UG004)

When these prerequisite steps are complete, the ACE GUI Snapshot Debugger view (see page 139) allows the evaluation /interaction with the running design in real-time.

The following sections further explain Snapshot and provide a guide through the process.

Accessing the Snapshot Debugger

Open the ACE GUI and Select the Project

Open the ACE GUI tool, and load or activate the selected project in the Projects View as shown below. See:

- Loading Projects, (see page 270)
- Setting the Active Implementation (see page 278)
- Working with Projects and Implementations (see page 268)

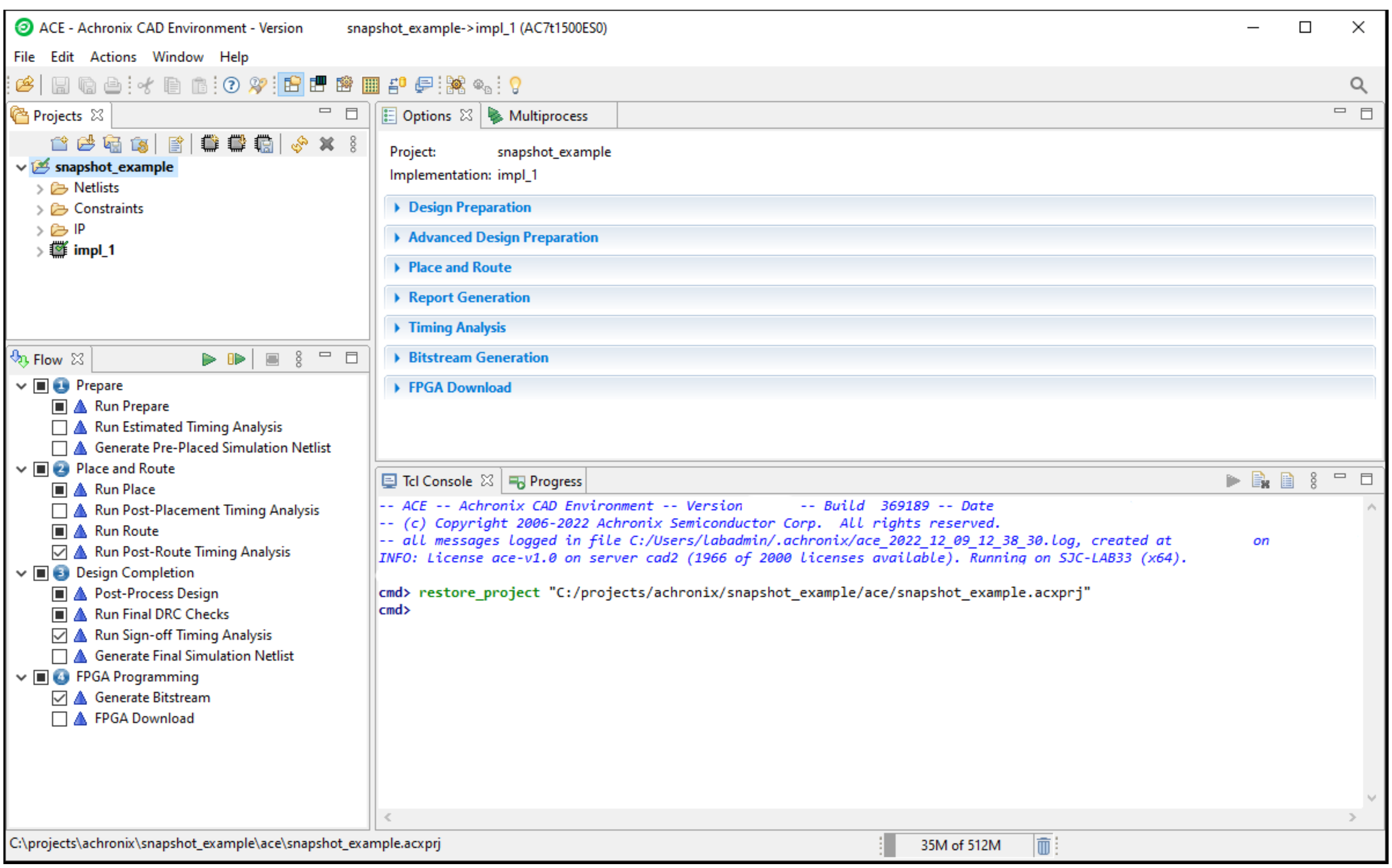


Figure 148: ***ACE Tool Load Project***

Open the Snapshot Debugger

Click the toolbar button to change to the ( ) Programming and Debug Perspective as described in the Working with Perspectives (see page 262) section. The Snapshot Debugger view (see page 139) should be visible by default, as shown below. If not, select **Window** → **Show View** → **Snapshot Debugger** from the main menu bar.

The Snapshot Debugger view (see page 139) should have automatically loaded the default Snapshot configuration file for the project, generated when the design ran through place and route, located in <ace_project_dir>/<active_impl_dir>/output/names.snapshot. If the file loaded, the correct signal names from the user design appear in the **Trigger Channels**, **Monitor Channels**, and **Stimuli** tables. If the file did not automatically load, click the ( ) **Load Snapshot Configuration** toolbar button in the Snapshot Debugger view (see page 139) to browse to the location of the preferred `*.snapshot` configuration file, or manually enter the signal names, channel widths, etc. to match the design.

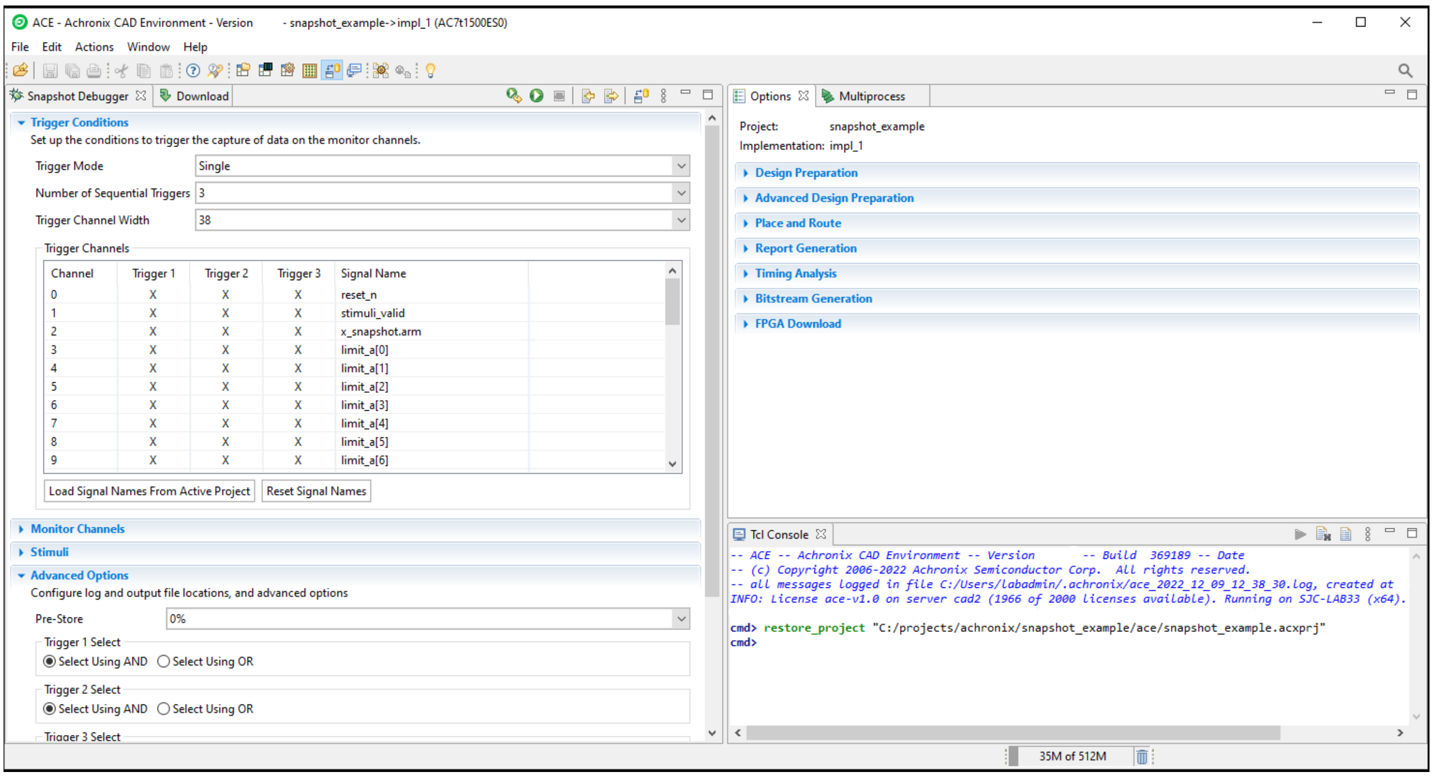


Figure 149: *Snapshot Debugger View*

Configuring the Trigger Pattern

Note

The Trigger Channel signal names are automatically configured to the correct values when the names.snapshot file is loaded. The `names.snapshot` file is generated during design preparation (the **Run Prepare** Flow Step (see page 221)), which contains the user design signal names connected to Snapshot, along with the trigger width and the maximum number of sequential triggers.

Configuring the Trigger Mode

The **Trigger Mode** option allows the user to select the trigger mode to use when the Arm action is run.

Single

The default trigger mode is **Single**, which means the trigger conditions are programmed in to the ACX_SNAPSHOT macro and then the GUI waits for a single trigger event to occur which matches those trigger conditions, and then a single VCD file is recorded. This option arms Snapshot and captures data only once.

Immediate

If **Immediate** trigger mode is selected, pressing the Arm button results in the same behavior as **Single** trigger mode, except that all 3 trigger patterns are treated as "Don't Care" (X's) so that the trigger event will occur as soon as the Arm button is pressed. This mode is useful to quickly capture the state of the running design without waiting for any trigger pattern to be met.

Repetitive

If **Repetitive** trigger mode is selected, the trigger conditions are programmed in to the ACX_SNAPSHOT macro and samples are captured repetitively until the upper limit of trigger event records is reached. When **Repetitive** trigger mode is selected, an additional set of repetitive trigger mode options will appear to allow the user to configure the number of sequential times Snapshot should be armed repetitively using the configured trigger conditions, and the way in which the output VCD files are managed. This mode is useful when the trigger conditions do not narrow in on the exact data pattern and the pattern you intend to observe occurs sporadically at the trigger conditions. You can let the repetitive trigger mode run for a long period of time, taking several capture records at the trigger conditions, to help find the pattern you are interested in. The user can optionally cancel the remaining Snapshot session once the desired data is captured.

The repetitive trigger Record Limit setting determines how many times (number of records) the GUI will repeatedly Arm the Snapshot debugger and capture samples. The user may set this to automatically run Snapshot up to 128 times.

The repetitive trigger VCD Record Limit setting determines how many Snapshot records to capture in a single VCD file. This essentially concatenates the VCD files from consecutive runs of Snapshot (records) into a single VCD file. The VCD file waveform contains a set of virtual signals to indicate the system timestamp at which each Snapshot record was captured. The user may concatenate up to 10 Snapshot records in a single VCD file.

If the Overwrite VCD File option is selected, the VCD Waveform File name specified in the Advanced Options section will be used to store the output VCD file. The file will be overwritten with the new VCD file each time the VCD record limit is reach. If the Overwrite VCD File option is not selected, then multiple VCD files will be written out and a unique VCD record number will be added to the VCD Waveform File name specified in the Advanced Options section for each VCD. For example, if you set the Record Limit to 8 and set the VCD Record Limit to 2, and set the VCD Waveform file path the "./snapshot.vcd", then Snapshot would output 4 VCD files to "./snapshot1.vcd", "./snapshot2.vcd", "./snapshot3.vcd", ". /snapshot4.vcd", each containing 2 Snapshot capture records.

Configuring Trigger Patterns

The Snapshot Debugger can be configured to use a **Trigger Channel Width** of 1 to 40 bits. The value entered in the Snapshot Debugger View must match the value of the *TRIGGER_WIDTH* parameter set on the ACX_SNAPSHOT module in the user design RTL. (This will be the width of the *i_trigger* bus.)

The Snapshot Debugger is capable of handling one to three sequential trigger patterns. The post-trigger data is sampled once the last trigger pattern in the sequence is matched.

The user may specify the number of desired sequential trigger patterns using the **Number of Sequential Triggers** option in the Snapshot Debugger View (see page 139). If **1** is selected, Trigger 2 and Trigger 3 are ignored. If **2** is selected, Trigger 3 is ignored and Snapshot will trigger when Trigger 1 is matched, followed (on any subsequent clock) by a match on Trigger 2. If **3** is selected, then Snapshot will trigger after a match on Trigger 1, followed (on any subsequent clock) by a match on Trigger2, followed (on any subsequent clock) by a match on Trigger3.

Each sequential trigger is hooked up to the trigger channels on the Snapshot Debugger core. The LSb of the trigger pattern is hooked to trigger channel 0, and the MSB is hooked to upper most trigger channel bit (TRIGGER_WIDTH - 1).

Each sequential trigger is made up of three parts: the pattern mask, the edge mask, and the don't care mask. In the Snapshot Debugger View, these 3 masks are combined for ease of use into a single trigger pattern value, which allows each bit to be specified as **X** (don't care), **R** (rising edge), **F** (falling edge), **0** (level 0), or **1** (level 1). The trigger pattern defines the trigger channel signal conditions that are required to detect a match. If a given trigger channel value is set to X (don't care), then this trigger channel is ignored when computing a match. If a given trigger channel value is set to R (rising edge), then this trigger channel is is evaluated as a match when a rising edge of this signal is seen by Snapshot. If a given trigger channel value is set to F (falling edge), then this trigger channel is is evaluated as a match when a falling edge of this signal is seen by Snapshot. If a given trigger channel value is set to 1 (level 1), then this trigger channel is is evaluated as a match as long as this signal's level is seen as a 1 by Snapshot (it is not edge sensitive). If a given trigger channel value is set to 0 (level 0), then this trigger channel is is evaluated as a match as long as this signal's level is seen as a 0 by Snapshot (it is not edge sensitive).

Warning!

If any active Trigger is configured with as all X's (don't care), the trigger pattern will be a match on the first clock cycle that trigger is evaluated.

The values within a trigger pattern may cause a trigger match event either by AND'ing or OR'ing. If AND'ing, then **all** signal values not masked (set to X) must match their pattern for the trigger match event to occur. If OR'ing, then the trigger match event will occur if **any** of the non-masked (not set to X) signal values match the specified pattern. The AND /OR configuration is set per sequential trigger using the **Select using AND** or **Select using OR** radio buttons. This selection can be different for each sequential trigger.

In the "Trigger Channels" table of the Snapshot Debugger View, the trigger patterns can be viewed and edited.

Setting Pattern Values Using the Table

For each channel, a value of **X** (don't care), **R** (rising edge), **F** (falling edge), **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under each "Trigger" column as shown below.

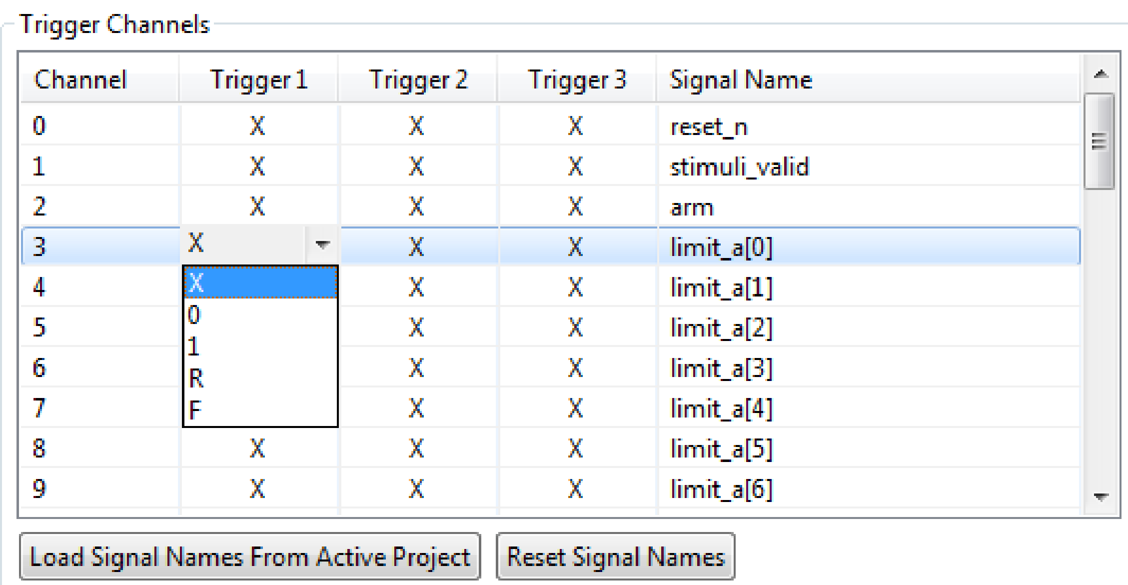


Figure 150: ***Trigger Channels Setting Example***

Setting Multiple Pattern Values as a Bus

The Assign Bussed Values Dialog wizard allows assigning a value to multiple signals from the Snapshot Debugger view (see page 139) "Trigger Channels" or "Stimuli Channels" tables as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all the selected signals in the Snapshot Debugger View (see page 139). There are 2 ways to launch this dialog to allow bus assignment of values:

1. With your mouse, left click to select a single row in the Snapshot Debugger View (see page 139) table which has a bussed signal name (i.e. din[2]). Then right mouse click to edit the **Value by Bus**. This method will automatically find all the other bits in the bus with the same signal name (i.e. din[0], din[1], din[2], etc.) and open the dialog to allow editing of the entire bus of signals.
2. With your mouse, hold CTRL or SHIFT and left click to select multiple rows in the Snapshot Debugger View (see page 139) table. Then right mouse click to edit the **Value by Selection**. This method will open the dialog to allow editing of all selected signals as a bussed value.

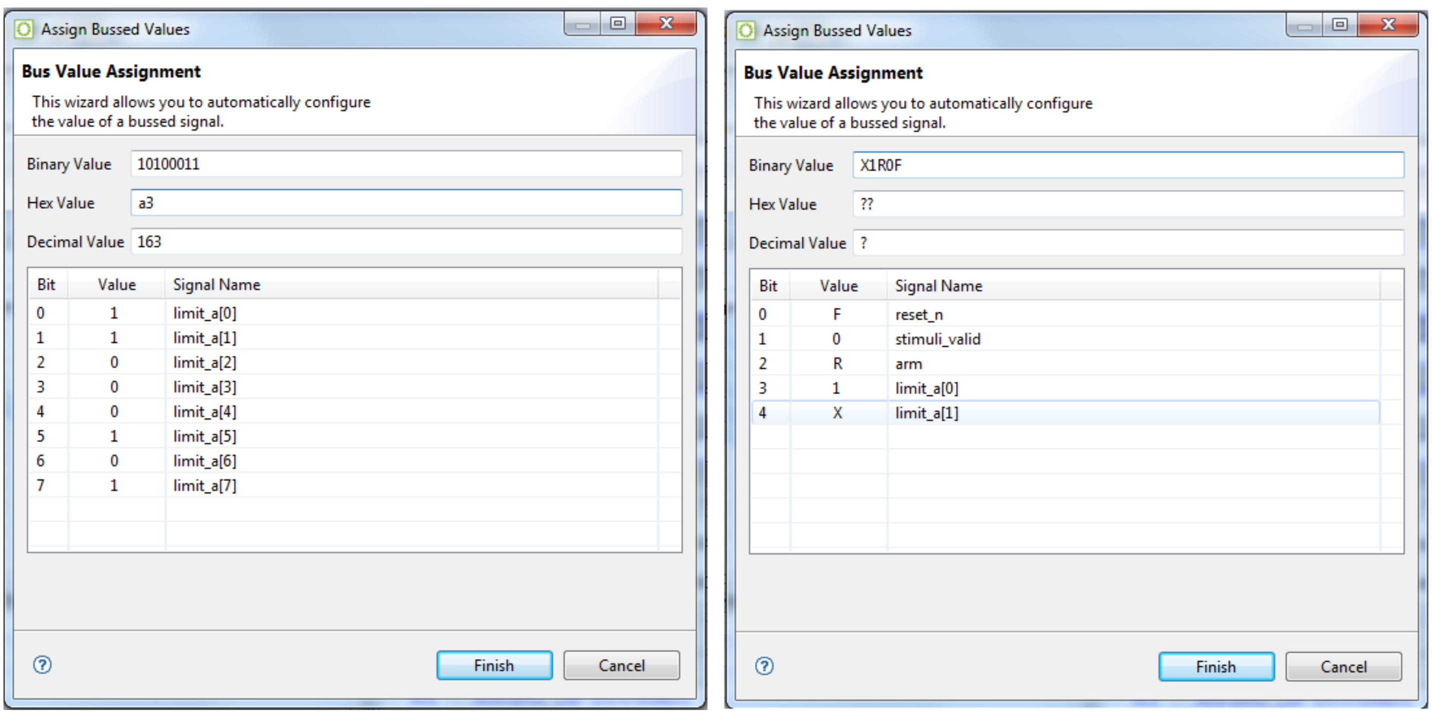


Figure 151: ***Assign Bussed Values Dialog Example***

See Assign Bussed Values Dialog (see page 151) for more information on this dialog.

Configuring the Monitor Signals

Note

The Monitor Signals are automatically configured to the correct values when the `names.snapshot` file is loaded. The file is generated during design preparation (the **Run Prepare**flow Step (see page 221)) which contains the user design signal names connected to Snapshot, along with the monitor width and number of samples.

The value of **Monitor Channel Width** in the SnapShot Debugger view (see page 139) must be configured to match the value of the `MONITOR_WIDTH` parameter of the `ACX_SNAPSHOT` instance inside the RTL of the design being debugged (this is the width of the `i_monitor` bus).

The value of **Number of Samples** in the SnapShot Debugger view (see page 139) should be configured to match the value of the `MONITOR_DEPTH` parameter of the `ACX_SNAPSHOT` instance inside the RTL of the design being debugged. If the value in the GUI does not match the value in the RTL, the value from the RTL is used and a warning is entered into the Snapshot log file.

Naming Captured Signal Data

Custom signal names for each channel can be entered under the **Signal Name** heading within the "Monitor Channels" table. The signal/bus names in the table are then used as labels on the captured signal data in the VCD waveform output, and are visible in the VCD Waveform Editor (see page 28).

Multiple signals can be combined into a bus by selecting multiple rows in the "Monitor Channels" table, right-clicking a selected signal row to bring up a popup context menu, and selecting ( ) **Assign Bus Name** from the context menu to bring up the Assign Bussed Signal Names dialog (see page 149). After configuring the bus in the dialog, the bus name and indices are propagated to all the previously-selected signals.

To select a contiguous range of rows:

1. Select the first signal.
2. hold the Shift key and select the last signal.

To select a non-contiguous set of rows:

1. Select the first signal.
2. While holding down the Ctrl key, select the other signals.

Signal names may be returned to their defaults by clicking the **Reset Signal Names** button under the "Monitor Channels" table.

> **Note**
>
> **Reset Signal Names** resets all signal names in the table at once, not just the currently selected rows/signals.
>
> The **Load Signal Names From Active Project** button loads the `names.snapshot` file generated during design preparation (the **Run Prepare** flow step (see page 221)) which renames all signals with their project-specific names, and also loads the project-specific default settings for monitor width, user clock frequency, default `.log` and `.vcd` file path, etc.

Configuring Test Stimulus

> The stimuli channel signal names are automatically configured to the correct values when the `names.snapshot` file is loaded. The `names.snapshot` file is generated during design preparation (the **Run Prepare** Flow Step (see page 221)), which contains the user design signal names connected to Snapshot, along with the stimuli width.

Snapshot has the capability to send 0 to 512 bits of test stimuli (the `ACX_SNAPSHOT` macro output signal `o_stimuli`) to the Design Under Test (DUT). This data is sent once per arming session, is only valid while the `o_stimuli_valid` signal is high.

This o_stimuli output is optional, and need not be connected to the DUT — it may safely be left floating when Snapshot is used to only read signals.

The value of **Stimuli Channel Width** in the SnapShot Debugger view (see page 139) must be configured to match the value of the STIMULI_WIDTH parameter of the ACX_SNAPSHOT instance inside the RTL of the design being debugged (this is the width of the o_stimuli bus).

In the **Stimuli Channels** table of the Snapshot Debugger View, the stimuli values can be viewed and edited.

Setting Stimuli Values Using the Table

For each channel, an output value of **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under the **Value** column as shown.

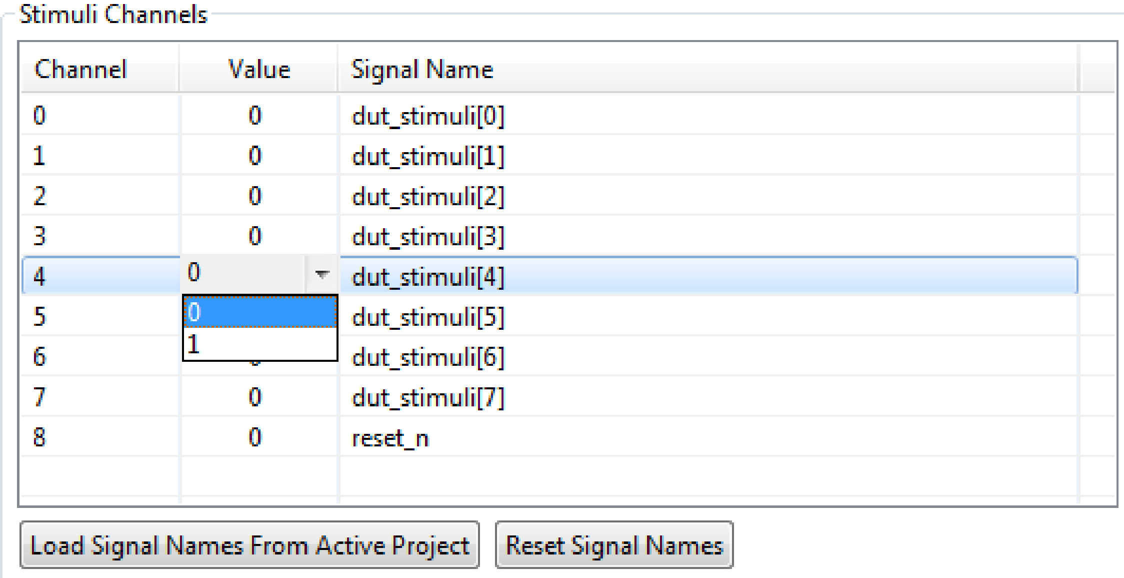


Figure 152: ***Stimuli Channels Value Setting Example***

Setting Multiple Stimuli Values as a Bus

The Assign Bussed Values Dialog wizard allows assigning a value to multiple signals from the SnapShot Debugger view (see page 139) **Stimuli Channels** table as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all the selected signals in the SnapShot Debugger View (see page 139). There are two ways to launch this dialog to allow bus assignment of values:

1. Left click to select a single row in the SnapShot Debugger View (see page 139) table which has a bussed signal name (i.e., din[2]).
   Right click to edit the **Value by Bus**. This method automatically finds all other bits in the bus with the same signal name (i.e., din[0], din[1], din[2], etc.) and opens the dialog to allow editing of the entire bus of signals.
2. Hold **CTRL** or **SHIFT** and left click to select multiple rows in the SnapShot Debugger View (see page 139) table.
   Right click to edit the **Value by Selection**. This method opens the dialog to allow editing of all selected signals as a bussed value.

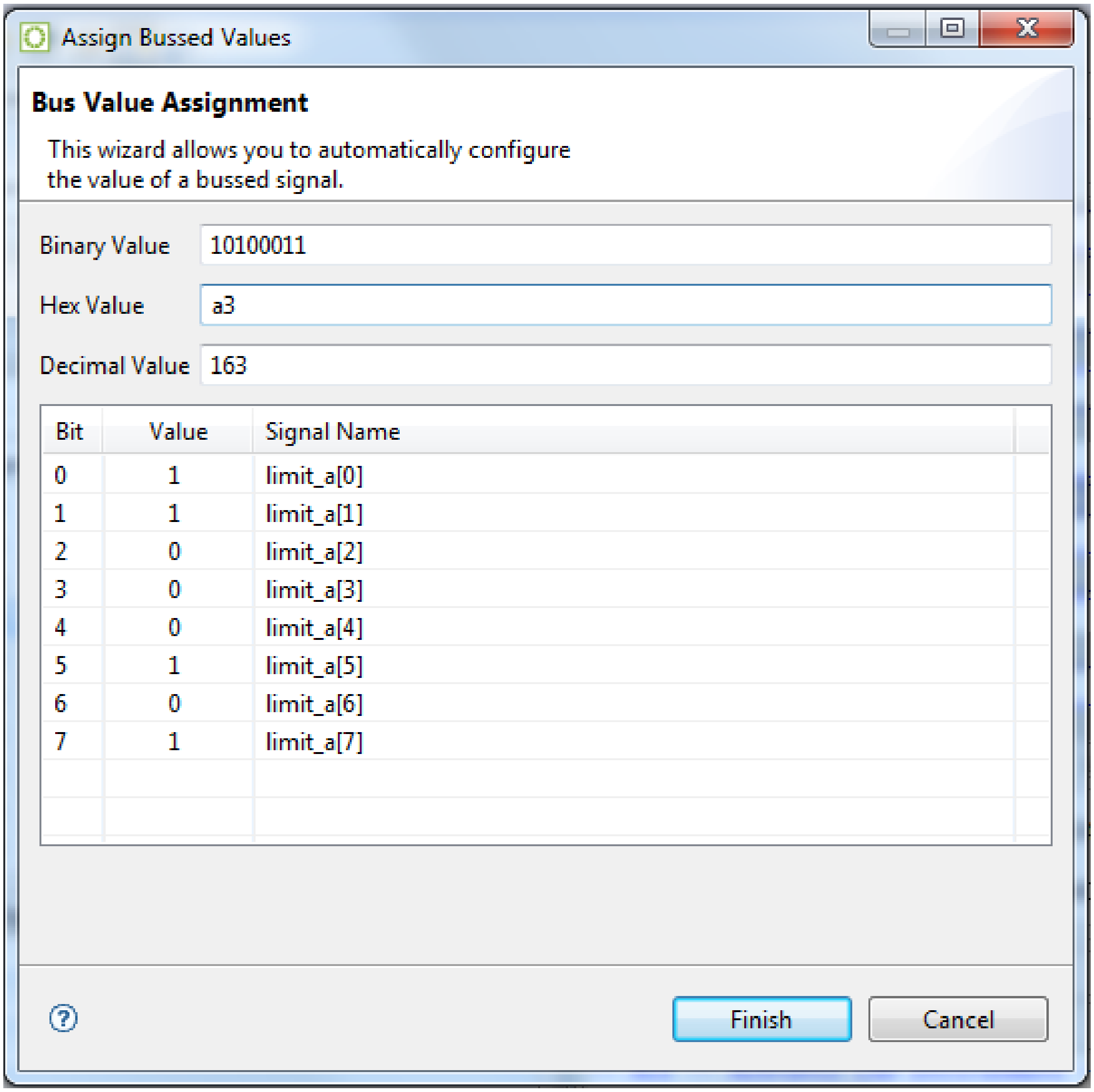


Figure 153: ***Assigned Bus Values Dialog Wizard Example***

See Assign Bussed Values Dialog (see page 151) for more information on this dialog.

Configuring Advanced Options

Pre-Store

In the Snapshot Debugger View (see page 139), the **Pre-Store** setting configures the portion of samples that are collected before the trigger, and (indirectly) how many are collected after the trigger.

For example, assume that Snapshot is configured to use a monitor depth of 1024 samples. See the table below:

Table 149: *Effect of "Pre-store" on samples collected before and after the trigger event*

"Pre-Store" value	Samples collected before trigger	Samples collected after trigger
0%	0	1024
25%	256	768
50%	512	512
75%	768	256

When a **Pre-Store** value other than **0%** is selected, the `.vcd` file contains a signal `snapshot_pre_store` that transitions (goes low) at the point where the (last sequential) trigger event occurred. Thus, the trigger event may easily be found without needing to actually count the samples.

Trigger Pattern Match Behavior

The values within a trigger pattern may cause a trigger match event either by AND'ing or OR'ing. If AND'ing, then *all* signal values not masked (set to X) must match their pattern for the trigger match event to occur. If OR'ing, the trigger match event occurs if *any* of the non-masked (not set to X) signal values match the specified pattern. The AND/OR configuration is set per sequential trigger using the **Select using AND** or **Select using OR** radio buttons. This selection can be different for each sequential trigger.

User Clock Frequency

The **Frequency** field must be configured to match the the `user_clk` frequency in the target user design, which typically matches the timing constraint set in the SDC file of the design being debugged. The value from the user design SDC file is set automatically in the `names.snapshot` file when an active implementation is available. The frequency value entered in the Snapshot GUI (or `.snapshot` configuration file) determines the time (in picoseconds) for all signals shown in the captured VCD file. All samples are captured at the rising edge of the Snapshot `user_clk` signal.

Configure Output File Locations

The final Snapshot configuration steps specify the locations of the output files which contain the log messages and sample data collected by Snapshot.

File Paths Relative To Chooses whether the **Log File** and **Waveform File** paths are understood to be relative to the **Active Project** directory or to the **Working Directory** (this only matters when the file paths provided are relative paths, and not absolute paths).

Log File configures the file name and path for the log file generated by the Snapshot Debugger run. The associated **Browse** button provides a directory/file selection dialog for the selection of a location different than the default (the default is `<active_impl_dir>/log/snapshot.log`, or if there is no Active Project and Implementation (see page 221), `<user_home>/snapshot.log`). If an error occurs during setup or while reading back the sample information, the Snapshot log file contains the error messages.

Waveform File configures the file name and path for storing downloaded sample waveform information from the SnapShot Debugger core in VCD format. The **Browse** button allows for the selection of a location different than the default (the default is `<active_impl_dir>/output/snapshot.vcd`, or if there is no active implementation, `<user_home>/snapshot.vcd`).

Collecting Samples of the User Design

Using the Startup Trigger

The Startup Trigger feature requires that the initial startup trigger parameters are configured on the `ACX_SNAPSHOT` macro to enable the Startup Trigger feature, and that the Arm Snapshot action has not been executed since the bitstream has been programmed. By clicking the ( ) **Capture Startup Trigger** button, the Snapshot Debugger view connects to the running `ACX_SNAPSHOT` circuit over JTAG and waits for the startup trigger condition to be met, retrieves the trace buffer contents, and outputs a VCD file. This feature is useful to capture trigger events that happen very soon after the Achronix FPGA enters user mode. When the ( ) **Arm Snapshot** button is clicked, the startup trigger conditions and any existing trace buffer contents are cleared. The Startup Trigger feature may only be used once after programming the bitstream.

Arming the Snapshot Debugger

When all the fields in the Snapshot Debugger view (see page 139) are configured, and the design is running on the target device, Snapshot is ready to be armed.

Select the **Arm** button (or the ( ) **Arm Snapshot** button in the SnapShot Debugger view toolbar), and the ACE Snapshot Debugger sends the configuration data (including the optional stimulus) to the `ACX_SNAPSHOT` circuit running on the Achronix device, waits for the trigger condition(s) to be met, retrieves the trace buffer contents, and outputs a VCD file as well as a LOG file.

When Armed, Snapshot begins to analyze the already-executing design in real-time.

The Snapshot log file and Snapshot waveform file are populated with the captured results, and the files are opened in ACE (the log file opens in an ACE text editor (see page 28), while the waveform (`.vcd`) file opens in the ACE VCD waveform editor (see page 28)). If an error occurs during Snapshot Debugger configuration or while reading back the sampled information (trace buffer), the Snapshot log file contains the relevant error messages, and the Snapshot waveform file is not created/updated.

The ( ) **Cancel** button aborts the Snapshot arming process. The Snapshot log file is updated, but the Snapshot waveform file is not created/updated if the cancel button is clicked. Cancel is useful if accidentally sending in trigger conditions that are never matched.

If using **Repetitive** trigger mode, Snapshot repetitively executes the arm action for the number of records specified, or until the cancel button is clicked. See Configuring the Trigger Pattern (see page 338) for details on the Repetitive Trigger feature.

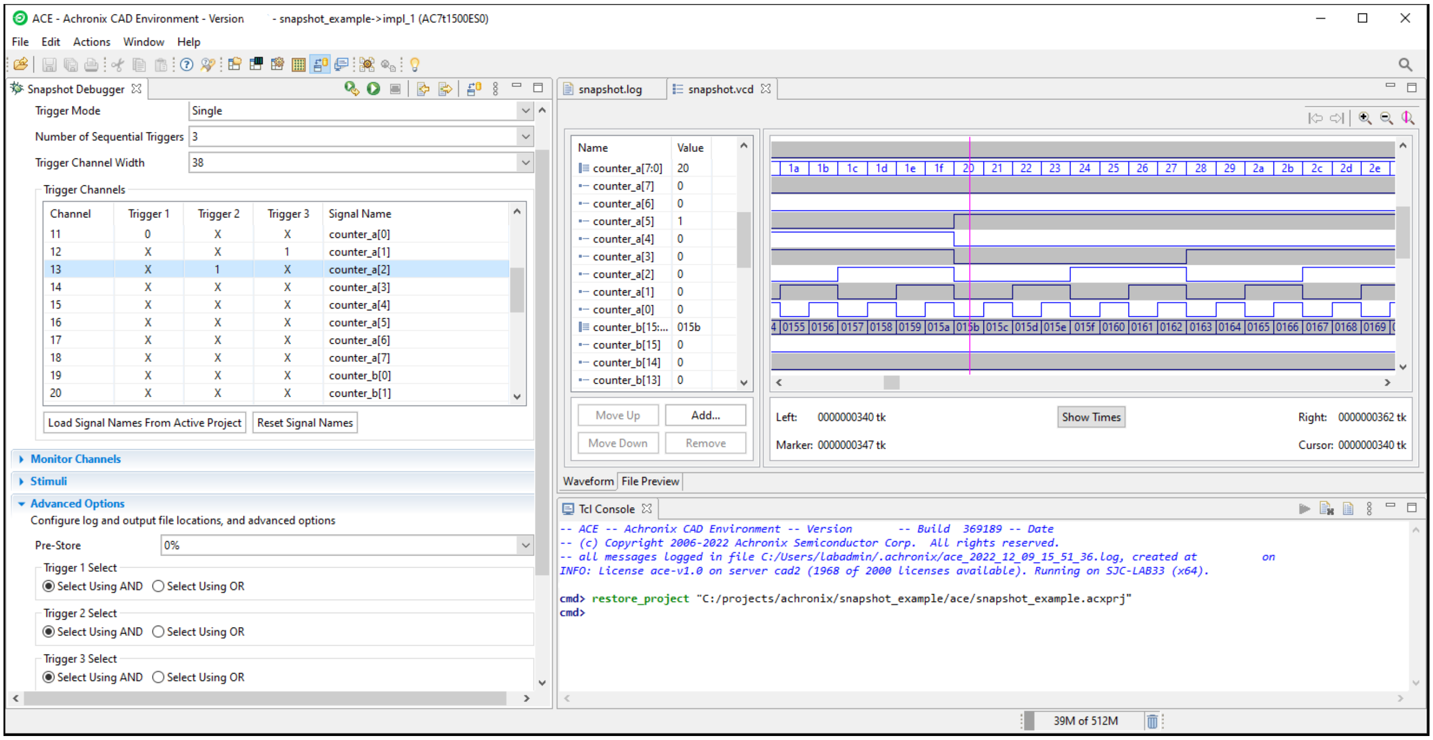


Figure 154: *Snapshot Debugger Arming Example*

Saving/Loading Snapshot Configurations

An existing known-good Snapshot configuration (the collection of settings in the Snapshot Debugger View (see page 139)) may be re-used at a later date, or in batch mode.

Snapshot configurations may be saved to a Snapshot configuration file (with the `.snapshot` file extension) using the ( ) **Save SnapShot Configuration** button found in the Snapshot Debugger view (see page 139) toolbar.

These Snapshot configurations may then be loaded later by using the ( ) **Load SnapShot Configuration** button, found in the Snapshot Debugger view (see page 139) toolbar.

> **Note**
>
> Previously saved Snapshot configuration files are necessary to run Snapshot in Batch mode (see page 348).

Tip

When a user design containing the `ACX_SNAPSHOT` macro completes the flow step (see page 221) **Run Prepare**, a `names.snapshot` configuration file is automatically generated. This file contains harvested information from the design including the monitor width, monitor depth, monitored signal names, trigger width, maximum number of triggers, trigger signal names, stimuli width, stimuli signal names, and user clock frequency. When an active project and implementation (see page 221) is available, the Snapshot Debugger view automatically loads the implementation `names.snapshot` file to pre-populate the relevant fields of the view. When generated, the file contains only a subset of a complete Snapshot configuration, and thus a generated `names.snapshot` file should not be used to drive Snapshot in batch mode (see page 348) via Tcl.

The names.snapshot configuration file can be loaded as a starting point to map the Snapshot RTL configuration into the Snapshot Debugger View. The Snapshot settings can be further customized and saved as custom Snapshot configuration files for later use.

Snapshot in Batch Mode

It is also possible to run Snapshot from ACE in batch mode. To do so, use the TCL command `run_snapshot`. Note that `run_snapshot` requires the use of a previously-saved (see page 347) Snapshot configuration file (`.snapshot`), and allows some values to be overridden from the TCL commandline. See the `run_snapshot` command in the TCL Command Reference section for further details.

The Snapshot configuration file may be edited manually in a text editor, or by configuring the Snapshot Debugger view (see page 139) in the ACE GUI and saving the Snapshot configuration (see page 347).

Example Snapshot Configuration File

```
#Snapshot Configuration File
#Tue Sep 12 13:52:54 PDT 2017
files_relative_to_project=1
frequency=322.0
log_file=./impl_1/log/snapshot.log
monitor_ch0.name=reset_n
monitor_ch1.name=stimuli_valid
monitor_ch10.name=limit_a[7]
monitor_ch11.name=counter_a[0]
monitor_ch12.name=counter_a[1]
monitor_ch13.name=counter_a[2]
monitor_ch14.name=counter_a[3]
monitor_ch15.name=counter_a[4]
monitor_ch16.name=counter_a[5]
monitor_ch17.name=counter_a[6]
monitor_ch18.name=counter_a[7]
monitor_ch19.name=counter_b[0]
monitor_ch2.name=arm
monitor_ch20.name=counter_b[1]
monitor_ch21.name=counter_b[2]
monitor_ch22.name=counter_b[3]
monitor_ch23.name=counter_b[4]
monitor_ch24.name=counter_b[5]
monitor_ch25.name=counter_b[6]
monitor_ch26.name=counter_b[7]
monitor_ch27.name=counter_b[8]
monitor_ch28.name=counter_b[9]
monitor_ch29.name=counter_b[10]
monitor_ch3.name=limit_a[0]
```

```
monitor_ch30.name=counter_b[11]
monitor_ch31.name=counter_b[12]
monitor_ch32.name=counter_b[13]
monitor_ch33.name=counter_b[14]
monitor_ch34.name=counter_b[15]
monitor_ch4.name=limit_a[1]
monitor_ch5.name=limit_a[2]
monitor_ch6.name=limit_a[3]
monitor_ch7.name=limit_a[4]
monitor_ch8.name=limit_a[5]
monitor_ch9.name=limit_a[6]
monitor_width=38
num_samples=4096
num_triggers=3
pre_store=0%
repetitive_trigger.overwrite_vcd=0
repetitive_trigger.record_limit=10
repetitive_trigger.vcd_record_limit=10
snapshot_version=3
stimuli=110010100
stimuli_ch0.name=stimuli[0]
stimuli_ch1.name=stimuli[1]
stimuli_ch2.name=stimuli[2]
stimuli_ch3.name=stimuli[3]
stimuli_ch4.name=stimuli[4]
stimuli_ch5.name=stimuli[5]
stimuli_ch6.name=stimuli[6]
stimuli_ch7.name=stimuli[7]
stimuli_ch8.name=do_reset
stimuli_ch9.name=stimuli_ch9
stimuli_width=9
trigger1=XXXXXXXXXXXXXXXXXXXXX00110101XXXXXXXXXXX
trigger1.select_using_and=1
trigger2=XXXXXXXXXXXXXXXXXXXXX1111R000XXXXXXXXXXX
trigger2.select_using_and=1
trigger3=XXXXXXXXXXXXXXXXXXXXXXXXXFXXXXXXXXXXXXXX
trigger3.select_using_and=1
trigger_ch0.name=reset_n
trigger_ch1.name=stimuli_valid
trigger_ch10.name=limit_a[7]
trigger_ch11.name=counter_a[0]
trigger_ch12.name=counter_a[1]
trigger_ch13.name=counter_a[2]
trigger_ch14.name=counter_a[3]
trigger_ch15.name=counter_a[4]
trigger_ch16.name=counter_a[5]
trigger_ch17.name=counter_a[6]
trigger_ch18.name=counter_a[7]
trigger_ch19.name=counter_b[0]
trigger_ch2.name=arm
trigger_ch20.name=counter_b[1]
trigger_ch21.name=counter_b[2]
trigger_ch22.name=counter_b[3]
trigger_ch23.name=counter_b[4]
trigger_ch24.name=counter_b[5]
trigger_ch25.name=counter_b[6]
trigger_ch26.name=counter_b[7]
trigger_ch27.name=counter_b[8]
trigger_ch28.name=counter_b[9]
```

```
trigger_ch29.name=counter_b[10]
trigger_ch3.name=limit_a[0]
trigger_ch30.name=counter_b[11]
trigger_ch31.name=counter_b[12]
trigger_ch32.name=counter_b[13]
trigger_ch33.name=counter_b[14]
trigger_ch34.name=counter_b[15]
trigger_ch4.name=limit_a[1]
trigger_ch5.name=limit_a[2]
trigger_ch6.name=limit_a[3]
trigger_ch7.name=limit_a[4]
trigger_ch8.name=limit_a[5]
trigger_ch9.name=limit_a[6]
trigger_mode=Single
trigger_width=38
vcd_file=./impl_1/output/snapshot.vcd
```

Playing a STAPL File (Programming a Device)

Warning!

The JTAG connection must be configured before using the Download View!
ACE interacts with the FPGA using the JTAG interface through a Bitporter pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the Download view. The configuration is managed using the Configure JTAG Connection Preference Page (see page 186), which is easily accessible by clicking the ( ) **Configure JTAG Interface** button in the Download view. See Configuring the JTAG Connection (see page 331) for more details.

A STAPL* bitstream file (`*.jam`) can be run or played from the Download view (see page 51). To access the Download view, change to the ( ) Programming and Debug perspective, or select **Window** → **Show View...** → **Others** → **Download View**.

From this view, individual STAPL Actions can be selected for playing (for example, the `PROGRAM` action to program the FPGA). The view also allows for any optional STAPL procedures within the chosen action to be selectively enabled /disabled.

* STAPL = Standard Test and Programming Language, JEDEC standard JESD-71.

Selecting a STAPL File

A STAPL bitstream file (`*.jam`) is selected under the "STAPL Design File" heading in the Download view (see page 51).

If the option for **Default File From Current Design/Impl** is selected, the filename/path field is made read-only and automatically populated with the default filename and path (the default file name is typically the name of the project or the name of the top module with the `.jam` file extension). If that default file does not exist, the "STAPL Actions and Procedures" tables display an appropriate error message.

The **Manual Selection** option allows manually choosing the path to a desired STAPL file from an arbitrary location:

- Click the **Browse** button to locate the file
- Enter a new full path
- Edit an existing path

- Select a previously used `*.jam` file from the editable file path combo-box by clicking the down arrow on the right side of the combo box

Lab Mode

When ACE is in Lab Mode, it is impossible to load designs (there is never a "current" design and implementation). Thus, the **Default File From Current Design/Impl** option is disabled, and the **Manual Selection**, option must be used. The file path combo-box still retains the last 15 files, to ease reuse.

Selecting Actions and Procedures to be Played

Under "STAPL Actions and Procedures", individual actions and procedures can be selected for playing. Pressing the **Refresh Lists from STAPL File Selected Above** button rereads the STAPL file, displaying each action and procedure contained in the selected STAPL file.

Under the heading "Action Name", an individual action can be selected to be played. Selecting an action causes all the procedures making up that action to be displayed in the procedures table:

- Each required procedure is automatically selected and cannot be deselected
- Recommended procedures are automatically selected, but can be deselected
- Optional procedures are automatically deselected, but can be selected to be run

When an action is played/run, only the selected Procedures are played/run. Deselected Procedures are skipped.

Table 150: *STAPL Procedure Execution States*

State	Icon	Description
(default)		These required procedures are always selected, and are always executed. Disabling these procedures is not allowed.
Recommended		These procedures are initially enabled, but disabling them is allowed.
Optional		These procedures are initially disabled, but enabling them is allowed.

Table Notes

- The checkbox icon appearance varies by operating system, window manager, and theme

Playing an Action

The selected action with the selected procedures can be run by clicking **Run '*action_name*' on the Connected Device**. The output is written to the Tcl Console View (see page 144) and saved in the ACE log file. Additionally, a log file of just the run itself is opened for viewing.

> **Caution!**
>
> When programming the FPGA using the Download view, the JTAG Scan Chain configuration specified on the Configure JTAG Connection Preference page (see page 186) overrides the JTAG scan chain configuration embedded in the STAPL file (the embedded configuration in the STAPL file was originally generated using the "Bitstream Generation" implementation options in the Options view (see page 103)).
>
> When using the `acx_stapl_player` from the command-line (instead of through the Download view GUI), the JTAG scan chain configuration embedded in the STAPL file is used instead, unless overridden with command-line arguments.

Optimizing a Design

There are numerous methods of design optimization available in ACE.

Many optimizations can be performed automatically by ACE, at the cost of additional runtime. These automatic optimizations are managed at a granular level through the implementation options (see page 215), which may be configured from the Options view (see page 103) and/or the `set_impl_option` Tcl command.

Achronix optimization experts have also collected together into option sets (see page 215) the implementation options which are known to work well together. These option sets may be used to create new implementations for user designs, to allow comparing/contrasting how various optimizations affect achieved frequencies and required runtimes.

Other optimizations must be performed manually, typically by editing the design source RTL or `.sdc` timing constraints. Analyzing critical paths (see page 324) is an important part of this process. Optimization through RTL changes is currently beyond the scope of this document — ask your Achronix Field Applications Engineer for more information regarding source optimization possibilities.

Attempting Likely Optimizations Using Option Sets

In addition to running multiple flows in parallel (see page 282), the Multiprocess view (see page 83) allows generating new implementations with auto-generated combinations of implementation options (see page 215). These known-good subsets of implementation options are called option sets (see page 215).

ACE can generate customized option sets based on design details (such as the target device) found within the active project and implementation (see page 221). These customized option sets are only generated by request, and are specific to the details of each implementation. To generate the customized option sets for the active implementation, use the **Refresh Option Sets** button in the Multiprocess view (see page 83), or the option `-create_option_sets` when calling the `run_multiprocess` Tcl command.

The Multiprocess view allows selecting a starting template implementation (see page 215) and then generating new implementations using the template implementation as a base. Each generated implementation overrides the implementation options found in the template implementation with the specified option set configuration (an overriding subset of the full collection of implementation options). The majority of the implementation options within the generated implementation are left with the same settings as existed in the template implementation. Only the options specified in the option set are overridden to take on new values. The newly generated implementation is given a name which includes the option set name for clarity (the generated name uses the template implementation name as a prefix, with the option set name as the suffix).

See the information in Running Multiple Flows in Parallel (see page 282), which discusses the basic use of the Multiprocess view (see page 83) and Multiprocess Summary report (see page 238) — the rest of this section builds upon those descriptions.

Selecting the Implementations to be Generated and Run in Parallel

After finding the Multiprocess view (see page 282), configuring the execution queues (see page 283), and configuring the desired flow to be followed by the selected implementations (see page 289), the implementations to be generated may be selected:

1. In the Projects view (see page 125), select (activate) the desired project (see page 215) and implementation (see page 215).
2. Select the radio button labeled **Generate Implementations from Option Sets**, found within the Multiprocess view Select Implementations (see page 86) section.
3. Click the **Refresh Option Sets** button to generate new option sets particular to the details of the active project /implementation. All of these custom option sets include "auto" in their names.
4. The Implementation Table within the Multiprocess view Select Implementations (see page 86) section is then updated to display a collection of potential implementations based upon the active implementation (see page 221), with names derived from the refreshed option sets.

The first entry in the Implementation Table is the active implementation itself. This implementation is the template from which all the generated implementations are derived. All other implementations in the table are generated, one for each option set, if they are selected (their checkbox is checked) when background execution is started. The Description column of the table indicates succinctly what implementation option changes are caused by each option set (thus describing how each generated implementation differs from the template, the active implementation).

Generating Option Set Implementations and Starting Background Execution

After the ( ) **Start Selected** button has been pressed, but before the behavior described in Starting Background Execution (see page 290) commences, ACE:

1. removes implementations in the active project with the same name as to-be-generated implementations
2. creates new implementations (exact copies of the template implementation) with the required names
3. applies the appropriate option set implementation options to the new implementations (overriding the inherited implementation option values with the subset making up the option set)
4. adds all selected (checked) implementations to the background processing queue(s), to be run through the flow

From this point on, the available functionality and behavior is identical to that described in Running Multiple Flows in Parallel (see page 282), beginning with starting background execution (see page 290).

Warning!

Each generated implementation which is selected overwrites *without prompting* any already-existing implementation with the same name in the active project. The template (active) implementation is not changed /overwritten. If a previously-existing implementation with a to-be-generated name collision is kept, the previously-existing implementation must be renamed to avoid the name collision *before* the ( ) **Start Selected** button is pressed.

Interpreting/Utilizing the Results

After viewing the results (see page 291), the final step of an optimization pass is usually comparing the results and choosing which generated option set implementation provides the best QoR in comparison to the template implementation.

By default, the implementations in the multiprocess summary report (see page 238) are sorted approximately by QoR, though it is likely still preferred to analyze the results in detail to choose which implementation is the best by preferred criteria.

> **Note**
>
> **Early Access Functionality**
> The automatic QoR sorting of the Multiprocess Summary report should be considered early access functionality. The sort details are likely to change (and improve) in future ACE releases.

That best generated implementation could then be renamed, so that it does not get overwritten by future multiprocess runs (e.g., it might be named "fastest1", "lowestpower1", etc.).

With the newly-renamed implementation selected in the Projects view (making it the active implementation), it also becomes the new template implementation in the Multiprocess view, ready for another multiprocess iteration through the option sets.

By iterating through several best template implementations (perhaps each with a new implementation name for "breadcrumb" purposes), the desired QoR may be reached.

> **Caution!**
>
> Ensure **Generate Implementations from Option Sets** is selected for each optimization iteration, otherwise any changed implementation options in the template implementation are not inherited by the option set implementations.
> Also, there is a scenario where all multiprocess results can be identical. The cause and a workaround are described in Running Multiple Flows in Parallel (see page 291).

Placement Regions and Placement Region Constraints

> **Warning!**
>
> Placement Regions and Placement Region Constraints are an advanced feature, and should only be used under the guidance of an Achronix FAE. Unguided use of placement region constraints can cause loss of QOR, or may make a design impossible for the Placer or Router to solve.

> **Note**
>
> **ACE automated placement often produces better QOR than user-defined placement regions/constraints**
> When attempting to use Placement Regions and Placement Region Constraints, it is highly recommended that a parallel implementation of the project lacking the user-defined Placement Regions be kept. In a number of tested cases, completely automated placement in ACE was able to produce better QOR than with user-defined Placement Regions and Placement Region Constraints. This can easily be achieved by keeping the placement region constraints in a separate pdc file, which can then be enabled or disabled for the place and route flow.

Placement Regions are user-defined rectangular areas of the core fabric (*not* the IO Ring), to which the user can inclusively constrain the placement of multiple instances from their design, without needing to manually assign instances to specific sites within that region.

Because of clock distribution limitations, only a finite number of clocks can be routed to each of the Clock Regions (see page 243) in the fabric. Placement regions allow advanced users to ensure that those constraints are met if the automated tools need guidance. When necessary, clocked instances (flops, BRAMs, etc) may be constrained to placement regions to guarantee ACE does not attempt routing more clocks into a region than the region can support.

Placement Regions and the associated instance placement constraints may be manipulated through Tcl, or via the ACE GUI using the Floorplanner View (see page 53) and Placement Regions View (see page 120). The Search View (see page 132), Selection View (see page 136), Critical Paths View (see page 48), and Netlist Browser View (see page 89) may also be used to assign instance placement constraints by using drag-and-drop operations.

Be aware that Placement Regions are not treated as distinct objects in the ACE design database. Thus, they do not have their own object type prefix (see page 305), nor are they directly searchable in the Search View (see page 132) or with the Tcl `find` (see page 553)command.

Placement Region Preferences

There are a number of user preferences which may be configured to alter how the mouse creates placement regions and assigns placement region constraints. These preferences are found on the Placement Regions Preference page (see page 208).

Creating a New Placement Region

Placement regions may be created/defined by using the mouse in the Floorplanner View, or by directly calling the Tcl command `create_region` . In both cases, the bounds of the created region may "snap to" (grow to encompass) the entirety of all enclosed Clock Region boundaries or tile boundaries.

To create a Placement Region using the mouse in the Floorplanner view:

1. Ensure the () Floorplanner Placement Region Tool is active.
2. (Optional) If the Placement region is meant to align with one or more Clock regions (see page 243), enable the overlays for those regions from the Clock Regions view (see page 36). This does not affect the functionality in any way, but makes it easier to know where to define the region bounds.
3. Press and hold the left mouse button with the cursor at one of the corners of the area to be defined as the new Placement region.
4. While still holding the left mouse button, drag the cursor to the opposite corner of the desired Placement region area. Release the left mouse button when the cursor reaches the desired location.
5. ACE calculates the enclosed subtile grid coordinates, growing the grid as necessary to ensure all partially-enclosed tiles are fully enclosed.
6. The Create Placement Region dialog (see page 159) pops up pre-populated with the calculated subtile coordinates:

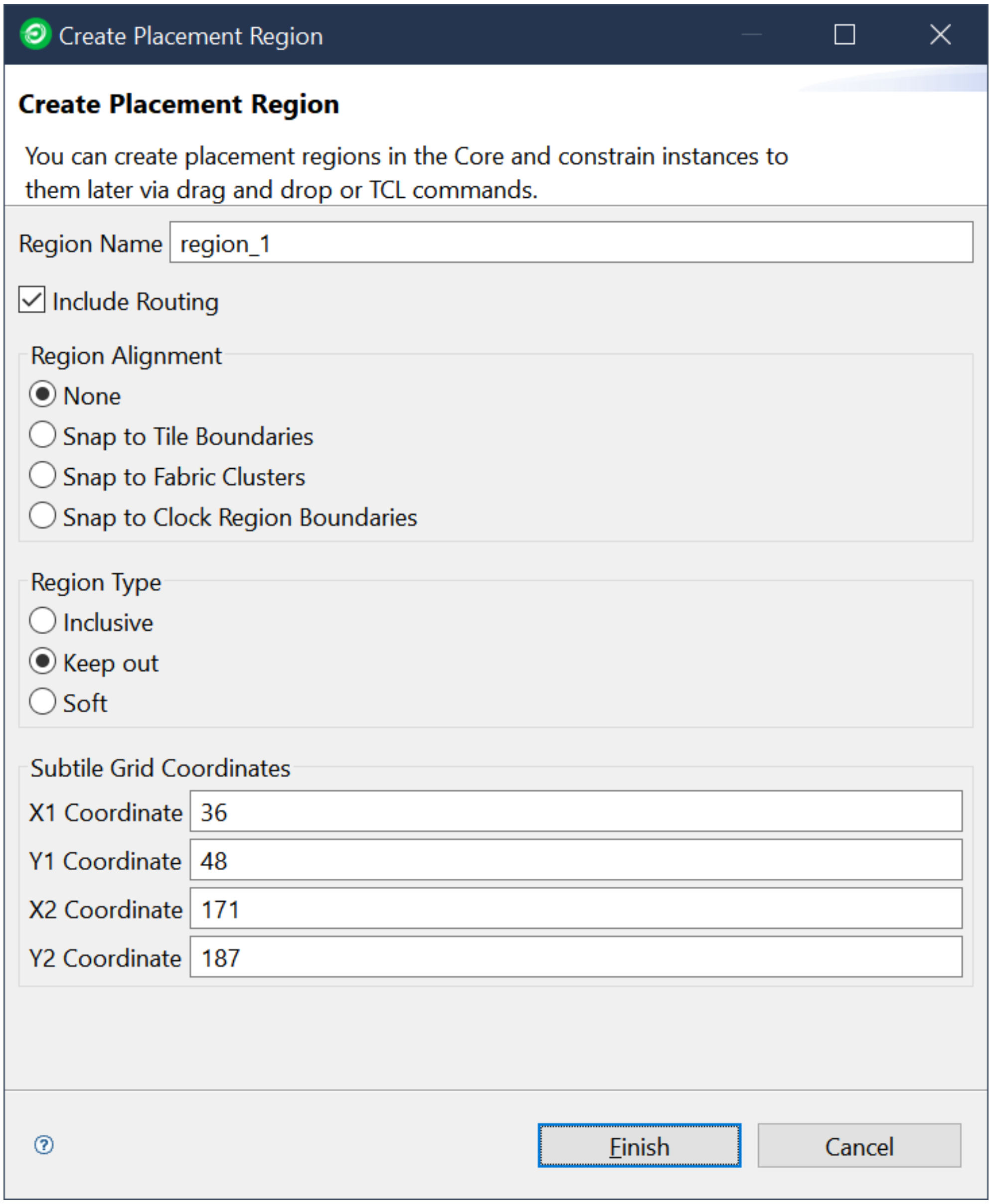


Figure 155: *Pre-Populated Create Placement Region Dialog*

7. Fill in the desired Placement region name.
8. Select whether the Placement region should be snapped to align with the edges of the Clock regions (see page 243), or the fabric clusters (see page 258), or with the more granular grid of basic resource tiles.
9. Select whether the Placement region should be an inclusive region, a "keep out" region, or a soft region (see the `create_region` Tcl command documentation for more information on these options).
10. Click the **Finish** button to create the new Placement Region.
11. ACE adds the new Placement Region to the table in the Placement Regions View (see page 120) and displays it as a translucent overlay within the Floorplanner (at this point, the region contains no constraints).

Resizing an Existing Placement Region

Existing Placement Regions may be resized with the `set_region_bounds` Tcl command, or with the mouse in the Floorplanner view. Any existing Placement Region Constraints for that region are kept — only the enclosed area is updated.

To resize a Placement Region with the mouse in the Floorplanner View (see page 53):

1. Ensure the ( ) Floorplanner Placement Region tool is active.
2. In the Placement regions view (see page 120), ensure the checkbox in the first column is selected for the desired Placement region. This makes the Placement region overlay visible within the Floorplanner view.
3. Ensure the **Snap To:** option in the Placement Regions Preference page (see page 208) is configured as desired.
4. (Optional) If the Placement region is meant to align with (snap to) one or more Clock regions (see page 243), enable the overlay for those regions from the Clock Regions view (see page 36). This action does not affect the functionality during the resize in any way, but makes it easier to know where to define the region bounds.
5. Move the mouse over any of the four corners of the placement region to be resized. The mouse cursor changes to a diagonal resize cursor when in a potential resize location.
6. Press and hold the left mouse button and drag the mouse to expand or shrink the Placement region area as desired.
7. Release the left mouse button when the mouse is at the desired location.
8. ACE calculates the enclosed subtile grid coordinates, growing as necessary to ensure all partially-enclosed subtiles (or Clock regions) are fully enclosed.
9. The Placement Region View table content is updated to show the new site counts enclosed by the Placement region, and the Floorplanner is updated to show the new Placement region overlay.

Moving an Existing Placement Region

Existing Placement Regions may be moved with the `set_region_bounds` Tcl command, or with the mouse in the Floorplanner view (see page 53). Any existing Placement Region Constraints for that region will be kept — only the enclosed area will be updated.

Be aware that the **Snap To** setting is enforced during the move — the enclosed area might not stay the same dimensions after the move. As with creating/resizing a region, the area will grow to ensure there are no partial sites in the enclosed area. It is frequently desired to resize (shrink) the Placement Region after a move, as it can easily grow larger than expected if sites/Clock Regions were partially enclosed at the ending mouse location.

To move a Placement Region with the mouse in the Floorplanner:

1. Ensure the ( ) Floorplanner Placement Region Tool is active.
2. In the Placement Regions view, ensure the checkbox in the first column is selected for the desired Placement Region. This selection makes the Placement Region visible within the Floorplanner view.
3. Ensure the **Snap To** option in the Placement Regions Preference Page (see page 208) is configured as desired.
4. (Optional) If the Placement Region is meant to align with (snap to) one or more Clock Regions (see page 243), enable the overlay for those regions from the Clock Regions view (see page 36). Enabling the overlay does not affect the functionality during the resize in any way, but makes it easier to know where to define the region bounds.
5. Move the mouse over the placement region to be moved. The mouse pointer changes to a "move" cursor when the mouse is over any placement region.
6. Hold the left mouse button while dragging the mouse to the desired new location for the placement region.

7. Release the left mouse button when the upper-left corner of the dragged region is at the desired location.
8. ACE calculates the enclosed subtile grid coordinates, growing as necessary to ensure all partially-enclosed subtiles (or Clock regions) are fully enclosed.
9. The Placement Region View table content is updated to show the new site counts enclosed by the Placement region, and the Floorplanner is updated to show the Placement region overlay at the new location (and with the latest dimensions).

Assigning Placement Region Constraints

Placement region constraints may only be assigned to core and boundary instances (not IO pads). Instances may be assigned placement region constraints interactively from the Tcl console, or from a PDC constraint file, with the `add_region_insts` and `add_region_find_insts` Tcl commands, or interactively with drag-and-drop mouse actions in the ACE GUI.

When using the `add_region_insts` or `add_region_find_insts` Tcl commands, the instances to constrain may be specified using an explicit list of instance names, or by clock domain name or critical path ID.

If specified as an explicit list of instance names the list may be formatted explicitly, or it may be the output of a `find` Tcl command.

```
add_region_insts "region_1" {i:inst1 i:inst2}
add_region_insts "region_1" [find -insts inst*]
add_region_insts "region_1" [find -insts inst* -filter {@type=DFF && @clock_domain=clk1}]
```

If specified by critical path ID, ACE determines which instances are part of that critical path, and assigns the placement region constraint to those Instances.

```
add_region_insts "region_1" {c:sc_s0}
```

Likewise, if specified by clock domain name, ACE determines which instances are part of that clock domain, and assigns the placement region constraint to all of those instances.

```
add_region_insts "region_1" {k:clka}
```

When writing PDC constraint files, the recommended practice is to use the `add_region_find_insts` command instead of the `add_region_insts` command. This is because:

1. When the instance list is specified with a `find` command, or by critical path ID or clock domain name expression, the command/expression is evaluated and expanded into a list at the time at which the `add_region_insts` command is evaluated (which happens at the *beginning* of the `run_prepare` flow step), not at the time at which it is applied with the `apply_placement` command (which happens at the *end* of the `run_prepare` flow step).
2. New instances which may be created during the `run_prepare` flow step, even if they would have matched the command/expression, are not included. Therefore, the `add_region_insts` command is best reserved for interactive use.
3. The `add_region_find_insts` command, on the other hand, specifies the `find` command as a string argument to be batched and evaluated later during the `apply_placement` command.

```
add_region_find_insts "region_1" "find -insts inst*"
add_region_find_insts "region_1" "find -insts inst* -filter {@type=DFF && @clock_domain=clk1}"
```

> **Note**
>
> **Saving Critical Path or Clock Domain Constraints**
>
> When critical paths or clock domains are used to specify the constraint, they are immediately expanded into a list of the corresponding instances within ACE at the time at which the `add_region_insts` command is evaluated. If the placement region constraints are later exported from ACE (and saved into a pdc file), they are exported as explicit lists of instance names and the original association with a critical path or clock domain is lost. More concise constraints for user designs may be created by manually entering the placement region constraint in the PDC file using the clock domain name instead of the list of explicit instances.

If any instance which was previously assigned a placement region constraint is assigned a new placement region, the prior constraint is overridden and discarded.

Optionally, placement region constraints may be restricted to allow only flops, in which case all other instances are excluded. (Setting these inclusion/exclusion preferences for mouse actions is done on the Placement Regions Preference page (see page 208).)

```
add_region_insts -flops_only "region_1" [find -insts * -filter {@clock_domain=clk1}]
add_region_find_insts -flops_only "region_1" "find -insts * -filter {@clock_domain=clk1}"
```

When placement region constraints are assigned to instances interactively using drag-and-drop mouse actions in the ACE GUI, the mouse drag-assign actions can start from:

- the Search view, where individual Instances and/or Paths, groups of Instances and/or Paths, or all Instances and /or Paths in the search results (if the titled branch nodes themselves are dragged, even the Instances/Paths not in the current set of 200 on the visible page of results) may be drag-assigned.
- the Selection view, where individual Instances and/or Paths, groups of Instances and/or Paths, or all Instances and /or Paths in the selection set (if the titled branch nodes themselves are dragged, even the Instances/Paths not in the current set of 200 on the visible page of results) may be drag-assigned.
- the Critical Paths view, where individual Paths or groups of paths may be drag-assigned.
- the Clock Domains view, where clock domains may be drag-assigned to include all applicable Instances from that clock domain in the assignment.
- the Netlist Browser view, where any node of the tree may be dragged, and all child nodes are included.

Mouse drag-assign actions can end at:

- An individual Placement Region row in the table within the Placement Regions View. After the assignment of the dropped Core/Boundary Instances completes, the site utilization counts are updated.
- A visible Placement Region overlay in the Floorplanner view, if the Placement Region Tool is active in the Floorplanner. After the assignment of the dropped Core/Boundary Instances completes, the site utilization counts in the Placement Regions view for that region are updated.

> **Note**
>
> **Overlapped Placement Regions**
>
> If multiple placement regions overlap visibly in the Floorplanner view, any Instances dropped within the visibly overlapping area are ignored. In such cases, instances must either be dropped in the Placement Regions view, or dropped in the Floorplanner view where there is no visible overlap (Placement Region overlays may be disabled from the Placement Regions view to eliminate visible overlaps — in these cases, constraint assignment occurs to whichever placement region remains visible at the Floorplanner drop location).

Listing all Objects Constrained to a Placement Region

The count of total sites of each type within each placement region is listed in the Placement Regions view (see page 120), along with the count of each Instance type for the sites.

If there are more instances constrained to a region than there are sites for that region, the corresponding cell in the Placement Regions view table turns red to indicate the problem.

To view a list in the Tcl Console view (see page 144) of all objects constrained to a placement region, do one of the following:

- Use the `get_region_insts` Tcl command.
- Right-click the desired Placement region in the Placement Regions view and select **Print Instances**.

Removing a Placement Region Constraint from an Object

Placement region constraints may be removed from individual core/boundary instances, or from all instances assigned to a region at once.

To un-assign a placement region constraint for individual core instances, use the `remove_region_insts` Tcl command.

To remove all instance constraints from a placement region, use the same Tcl command, or in the Placement Regions view (see page 120), right-click the desired placement region, and select **Clear Placement Region**.

Saving Placement Region Definitions and Placement Region Constraints

Placement region constraints may be saved:

- from the Floorplanner view (see page 53) with the "Save Pre-placement Constraints" action (which displays the Save Placement dialog (see page 170))
- from the Placement Regions view (see page 120) with the "Save Placement Regions" action (which displays the Save Placement Regions dialog (see page 172))
- by using the `save_regions` Tcl command directly

Note

Important Consideration When Saving Placement Region Constraints

Only the final list of all individual instances being constrained is saved. The individual Tcl commands which built up the final list of constraints (including "find" commands, the extraction of instances from critical paths, or from clock domains) is lost. The saved PDC file may be edited to replace explicit lists of instances with `find` commands or clock domain names.

Deleting Placement Regions

Unwanted Placement Regions may be deleted from the Placement Regions view (see page 120) by right-clicking the region in the table and selecting **Remove Placement Region**.

Alternately, the `remove_region` Tcl command may be called directly.

Running the HW Demo

The HW Demo facility is primarily intended as an aide to Achronix field application engineers (FAEs) that allows them to conveniently demonstrate particular features of Achronix FPGAs. Demonstration designs built into the ACE GUI software can easily be loaded into the attached board/device and executed. As the demonstration design is executing, the status of the design can be monitored in real-time, and visually represented within the HW Demo display.

The HW Demo facility uses fully functional designs (not included within an ACE installation, but provided as directory overlays) to demonstrate the real world application of hardened IP blocks. A given design may consist of a single IP block type, but typically they combine several IP block types working in a coordinated manner. These prebuilt designs are also useful to new ACE users as a way to gain experience setting up the Bitporter and prototyping environments.

Installing HW Demo Designs

Each HW demo or reference design (including bitstreams, additional software, documentation, and source files when possible) is packaged into and delivered in a single tarball, ZIP, or Windows installer file, downloadable from the Achronix support site. A set of installation instructions is provided as a separate document (not here) as the details may vary for each design. Installation might require several steps, depending on the software tools and drivers needed.

There are expected to be two types of designs available. Reference designs are meant to be modified, while demo designs are black boxes. Reference designs typically are installed into the user home directory (to encourage editing), while demo designs may be installed into the <ace_install> directory (which often has read-only permissions to discourage accidental overwrites). Both design types use the same framework within ACE, and both are presented through the HW Demo view in the ACE GUI.

Ask your FAE for further details about acquiring and installing the HW demo and reference designs for your specific development kit.

HW Demo Installation Paths

By default, when ACE starts up, it searches for installed HW demos in the following paths:

- <userhome>/achronix/ref_designs/
- <ace_install>/ref_designs/

After downloading a design tarball or zip, the design should be unpacked into either of those directories.

Selecting The Target Device And Demo

At the top of the HW Demo view (see page 64) are controls for selecting the target device and an associated demonstration design. After selecting a target device (or the default device matches the device you are working with) the list of available designs is accessible in the **Demo Design** control.

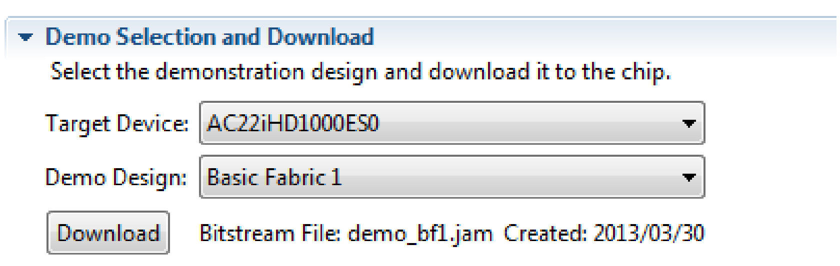


Figure 156: ***Demo Design Control Example***

Note

If no demos are installed, these controls remain disabled (which indicates the lack of installed designs).

Loading The Demo JAM File

After selecting the target device and demonstration design, the name of the `*.jam` file appears to the right of the **Download** button. Click the **Download** button to initiate loading of the the design into the attached FPGA device. Any designs that are running when **Download** is clicked are terminated without warning. If there are any errors or problems during the download process, a pop up dialog will be displayed with an explanatory message. When the selected design has been loaded and started, monitoring of the attached FPGA device is initiated using the DCC connection.

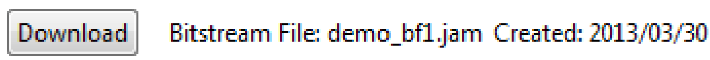


Figure 157: ***Demonstration Design Download Example***

Displaying Board Status

After a design has been loaded and started running, ACE may monitor the status of the demonstration board LEDs and DIP switches, as well as key internal conditions such as core voltage, temperature, etc. Clicking the visualization of an LED in the HW Demo view causes the corresponding actual LED (on the demonstration board) to toggle state.

Note

The visualized DIP switches are only used for reporting the state of the corresponding actual switch on the demonstrations board. The physical DIP switch cannot be set by clicking its image in ACE.

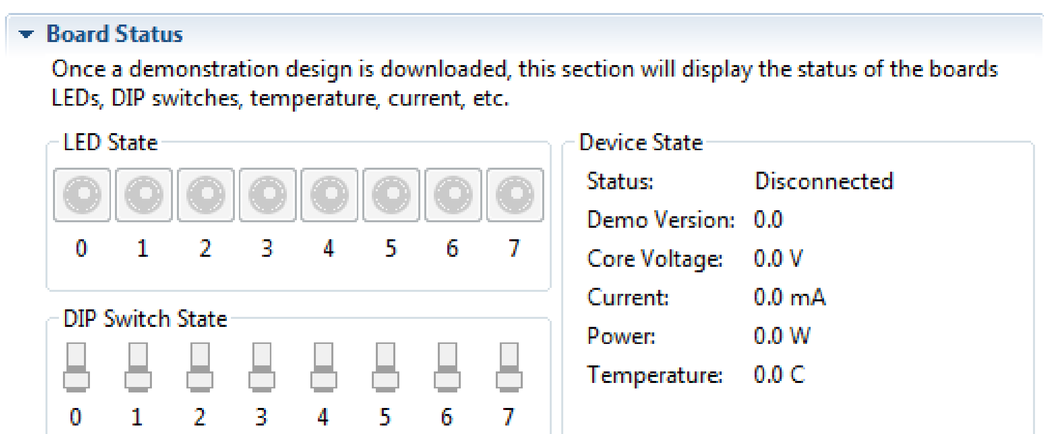


Figure 158: ***Rudimentary Demo Design Example***

Control of Running Demonstration Design

While the Snapshot Debugger has extensive facilities for collecting data samples from a running design, it does not currently provide any direct mechanisms for controlling or interacting with a design. The HW Demo view (see page 64) may provide a simple set of on-screen controls for reading and writing register values in some demo designs. In a demo similar to the example shown below, to read a register value, enter its address and click the **Read** button. The current value of the specified register appears in the **Data:** field to the right. Likewise, to modify a register value, enter its address and new value in the provided fields, and click the **Write** button.

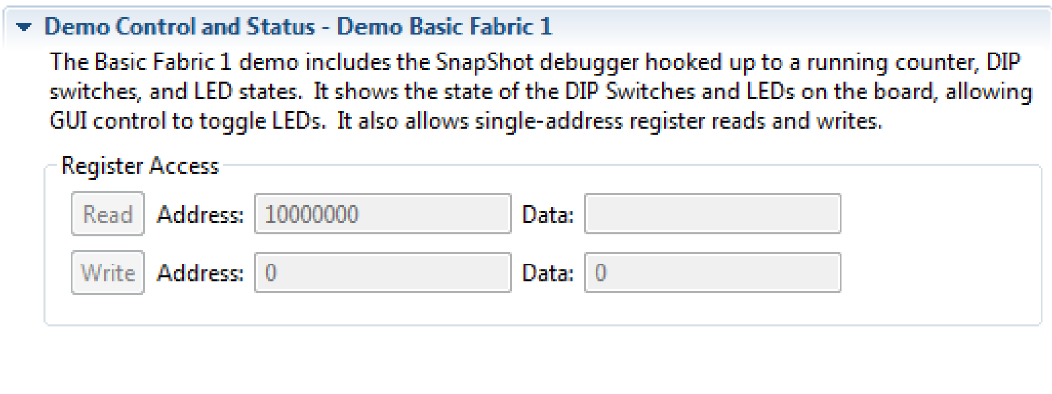


Figure 159: ***HW Demo View Example***

Using Incremental Compilation (Partitions)

This section begins with a high-level overview, and then continues with detailed tutorials.

Overview of Incremental Compilation and Partitions

Upstream synthesis tools have the ability to break a design up into smaller logical units (see: *Synplify Pro for Achronix User Guide*, Chapter 11: Working with Compile Points). Within ACE these smaller logical units are called "Partitions". These partitions can each be thought of as a nearly independent block — each partition can potentially be synthesized, optimized, placed, and routed independently. Because of this independence, when only one partition changes, only that partition needs to be re-run through the flow, leading to a significant runtime savings.

Defining Partitions

It is expected that partitions are defined primarily by the upstream synthesis tool. The synthesis tool typically exports a partition definition/constraint file. For example, the file below is an example of a `*.prt` file exported by Synplify Pro for Achronix.

Example partition definition (*.prt) file

```
set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1476335212" -cp_type "hard"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp
"1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out" -view "fpu_out" -timestamp "1476335212" -
cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp
"1476335212" -cp_type "locked"
```

```
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div" -view "fpu_div" -timestamp "1476335212" -
cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -
timestamp "1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp
"1476337611" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1476337611" -
cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp
"1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in" -view "fpu_in" -timestamp "1476335212" -
cp_type "locked"
```

Enabling Incremental Compilation

Enabling incremental compilation support within ACE is quite easy, assuming the partitions are already defined through the upstream synthesis tool:

1. In the Projects view (see page 125), add the partition definition file(s) to the ACE project (see Adding Source Files (see page 272)). The new partition definition file appears in the Projects view as a **Constraints** file and, in the Options view (see page 103), in the **Design Preparation** section in the list of **Constraints Files** (and should already have its checkbox selected).
2. In the Options View (see page 103), within the **Design Preparation** section, select the checkbox labeled **Enable Incremental Compile**.
3. In the Projects View, save the current project (see page 269).

From this point forward, this Project/Implementation uses incremental compilation when running the flow.

Note

The presence of the partition definition constraint file in the project, plus the checked **Enable Incremental Compile** implementation option, are the only configuration changes that distinguish the incremental compile flow from the standard non-incremental flow.

Tracking Partition Status

ACE provides two main tools for checking the compilation state, timestamps, and other statistics of each partition.

Partitions Report

The Partitions report (see page 228), automatically generated (and opened in the GUI) during the **Run Prepare** flow step (see page 221), shows the current status of each of the partitions, including resource counts and re-compilation states.

Partitions View

Similar to the Partitions report, the Partitions view (see page 117) shows the status of each partition and a variety of other statistics. Additionally, the view allows for ease of visualization of the partitions and their relationships to the instances and each other.

Forcing an Unchanged Partition to Recompile

When using the Partitions view, ACE provides a mechanism to override the partition timestamp during the next pass through the flow (see page 221). The column named **Force Re-compile on Next Run** displays the status of this override mechanism.

To mark a partition as needing forced compilation:

1. Click the partition in the Partitions view.
2. Right-click anywhere in the partition row to open the context menu, and choose **Force Partition Changed**.

A check mark appears in the **Force Re-compile on Next Run** column of the view in the row containing the partition. The next time the flow is executed, the partition is re-placed and re-routed, even if there were no RTL changes and it was not re-compiled in the upstream synthesis tool.

To remove the mark for forced recompilation:

1. Click the partition in the Partitions View.
2. Right-click anywhere in the partition row to open the context menu, and choose **Un-Force Partition Changed**.

The check mark disappears in the **Force Re-compile on Next Run** column of the view in the row containing the partition. The next time the Flow is executed, the partition is only re-placed and re-routed if the partition was re-compiled in the upstream synthesis tool.

Note

The ACE forced recompilation flag is a one-time trigger. When compilation is completed, any force flags for that implementation are cleared.

Tip

Forcing all Partitions to Re-compile

The easiest way to force all partitions to immediately be recompiled (run through the entire flow) is:

- Enter the following Tcl command in the Tcl Console view (see page 144):

```
run -ic init
```

- Alternately:
  1. Change to the Projects Perspective.
  2. In the Flow view (see page 61), enable and disable the optional Flow steps (see page 221) as desired.
  3. Right-click any flow step, and select the context menu item **Re-Run Flow with "-ic init"**.

See Running the Entire Flow (see page 280) for additional details.

Viewing Instances In Partitions

There are multiple ways to quickly see which instances belong to a given partition:

- The Search view (see page 132) and the `find` Tcl command can both be used to list all the instances within a partition or list of partitions, using the `@partition` filter. The Search Filter Builder dialog (see page 174) might ease the building of the filter for the Search view.
- Adding all the instances within a partition to the ACE Selection set (using the Selection view (see page 136), especially when populated with search results) is an easy way to see where members of a partition are within the Floorplanner view (see page 53). When the Floorplanner layer for **Selected Instance Flylines** is enabled, the connectivity of the selected partition is also visible.

- The Netlist Browser view (see page 89) is a table of the instances (and enclosing macros) making up the design, with a column indicating the partition for each instance. This table can be filtered by column values, thus the table can be filtered to include only the instances within a given partition.
- Using highlight colors assigned from any of the above views (especially using the Partitions view **Auto-Highlight** functionality) can make it easy to see how members of various partitions are placed in relation to each other in the Floorplanner.

The Floorplanner view also includes a new color in the Instance states (see page 243) for the new "Locked" state relating to partitions. Instances that are locked are a member of a locked partition that has remained unchanged since the prior incremental compilation. ACE does not change the site assignment for that instance during the Placement phase of place-and-route.

Related Tcl Commands

The following Tcl commands were created specifically to interact with partitions:

- `get_partition_changed` (see page 561)
- `get_partition_force_changed` (see page 561)
- `get_partition_info` (see page 561)
- `get_partition_insts` (see page 562)
- `get_partition_timestamp` (see page 562)
- `get_partition_type` (see page 563)
- `is_incremental_compile` (see page 570)
- `report_partitions` (see page 579)
- `set_partition_force_changed` (see page 602)
- `set_partition_info` (see page 603)

Additionally, the following Tcl commands were enhanced with additional options specific to incremental compilation and /or partitions:

- `run` (see page 585)
- `filter` (see page 551)
- `find` (see page 553)

Incremental Compile Tutorial

Overview

This tutorial demonstrates the process of running incremental design compile within ACE. This tutorial consists of two parts:

- Single-Process Incremental Compile Tutorial (see page 367) – covers how to process a single-pass incremental compile. This first tutorial must be run before the Multiprocess Incremental Compile tutorial
- Multiprocess Incremental Compile Tutorial (see page 407) – details how to run a set of changes in order to select an optimal implementation. This second tutorial expands upon concepts from the first and cannot be run standalone.

Tutorial Files

> **Note**
>
> The files needed for this tutorial are no longer available due to the their use of obsolete components. Despite this fact, the procedure outlined in the tutorial remains valid and is presented here to provide a detailed overview of the incremental compile procedure.
>
> This is an advanced tutorial. It assumes that both Synplify Pro and ACE are installed in your system search path and that you are already familiar with the use of both tools. If that is not the case, start with an introductory tutorial for those tools.

Single-Process Incremental Compile Tutorial

The goal of this tutorial is to illustrate the incremental compile flow from an initial version of RTL, through the following steps:

1. Synthesis in Synplify Pro.
2. ACE place and route.
3. A modification of the original RTL.
4. Back through the flow.

The goal of the flow is to help minimize the time it takes to make incremental changes to existing RTL and get those changes through ACE with the minimum amount of time and perturbation to the existing design implementation in ACE.

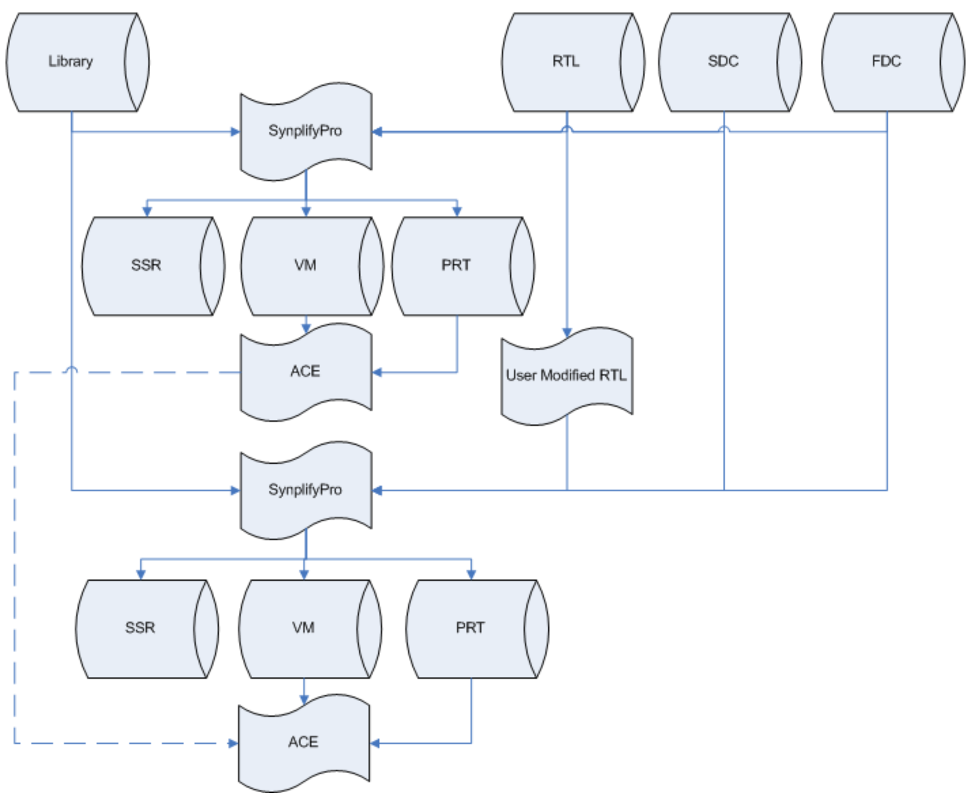


Figure 160: ***Incremental Compile Flow Chart***

Note

Legend

= Step that is integral to the running of this tutorial.

= Items that should be checked at this point in the tutorial to gain insight in to how the flow works and the feedback the tools are providing.

Bold Text = Text that can be found as a label to some GUI component including report table headings.

Step 1: Obtaining the Files

The tutorial reference design ZIP file is no longer available. Please refer to the following tutorials for procedural details using you own design files.

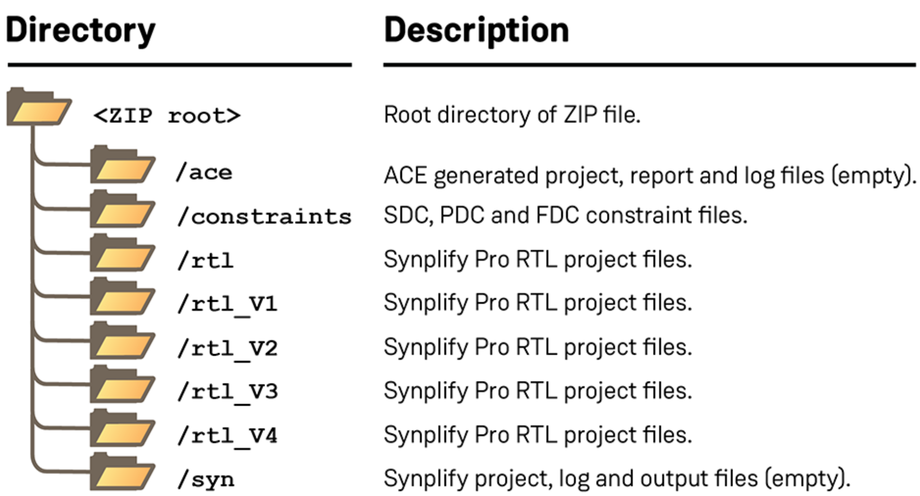


Figure 161: ***Tutorial Directory Structure***

Step 2: Set up the Synthesis Project

Start the Synplify Pro GUI. For Linux:

```
% cd <your work area>/Speedcore_Incremental_Compile_RefDesign_RD012/synplify
% synplify_pro
```

For windows, double-click the Synplify Pro Icon.

Create a new project with ( ) **Open Project** → **New Project** (or **File** → **New Project** in Windows). Windows users need to ensure that the project is saved to the chosen work area (**File** → **Save As**). To follow the directory structure used in this tutorial, use <your_work_area>/Speedcore_Incremental_Compile_RefDesign_RD012/synplify. The Synplify Pro home screen appears with an empty project named `proj_1` and an implementation named `rev_1`, as in the following screen shot:

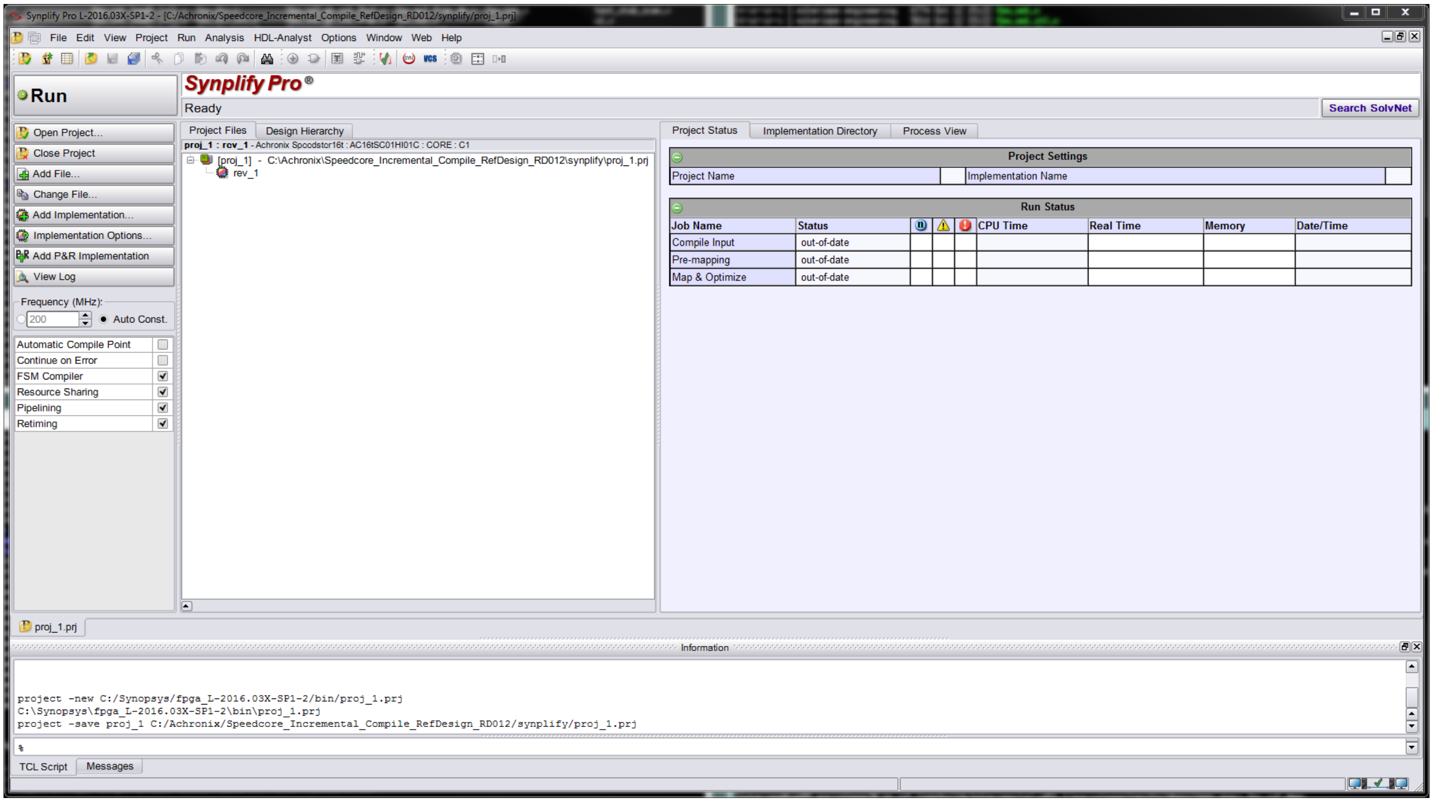


Figure 162: *Synplify Pro Home Screen*

Select **Project** → **Add Source File** to bring up the **Add Files to Project** dialog box. click the blue ( ) button to navigate up to the parent directory. In the "Files of type:" combo-box, select **All Files (*)**. Then double-click `constraints` to navigate to that directory and click the **<-Add All** button to add all of the constraint files. Click the blue up-arrow to navigate up one level and add all of the Verilog files in the `rtl` directory; click **OK**.

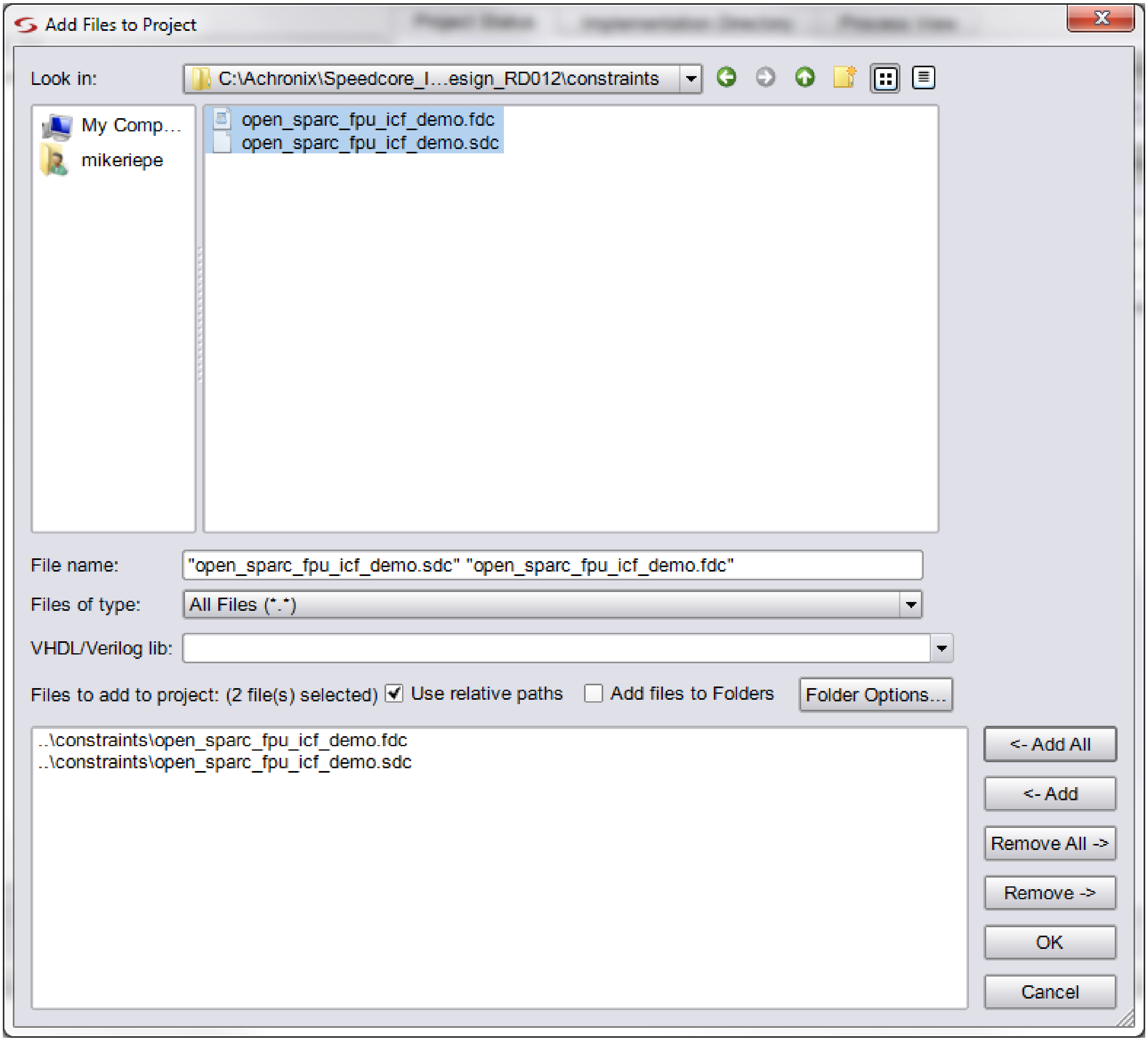


Figure 163: ***Add Files to Project Dialog Box***

All of the files just added are now listed under the `proj_1` project in the Project Files tab (click **+** to expand each file type).

Ensure that the file `fpu_top.v` (the top-level module) appears last in the list of Verilog files. If it does not, correct the order by using the mouse to select the file name and drag it to the end of the list. For Windows users, the Result Base Name is set to the project name used earlier, the default being Proj_1. To have the file names match this document exactly, manually change the **Project** → **Implementation Options** → **Implementation Results** → **Result Base Name** to "open_sparc_fpu_icf_demo", and click **OK**.

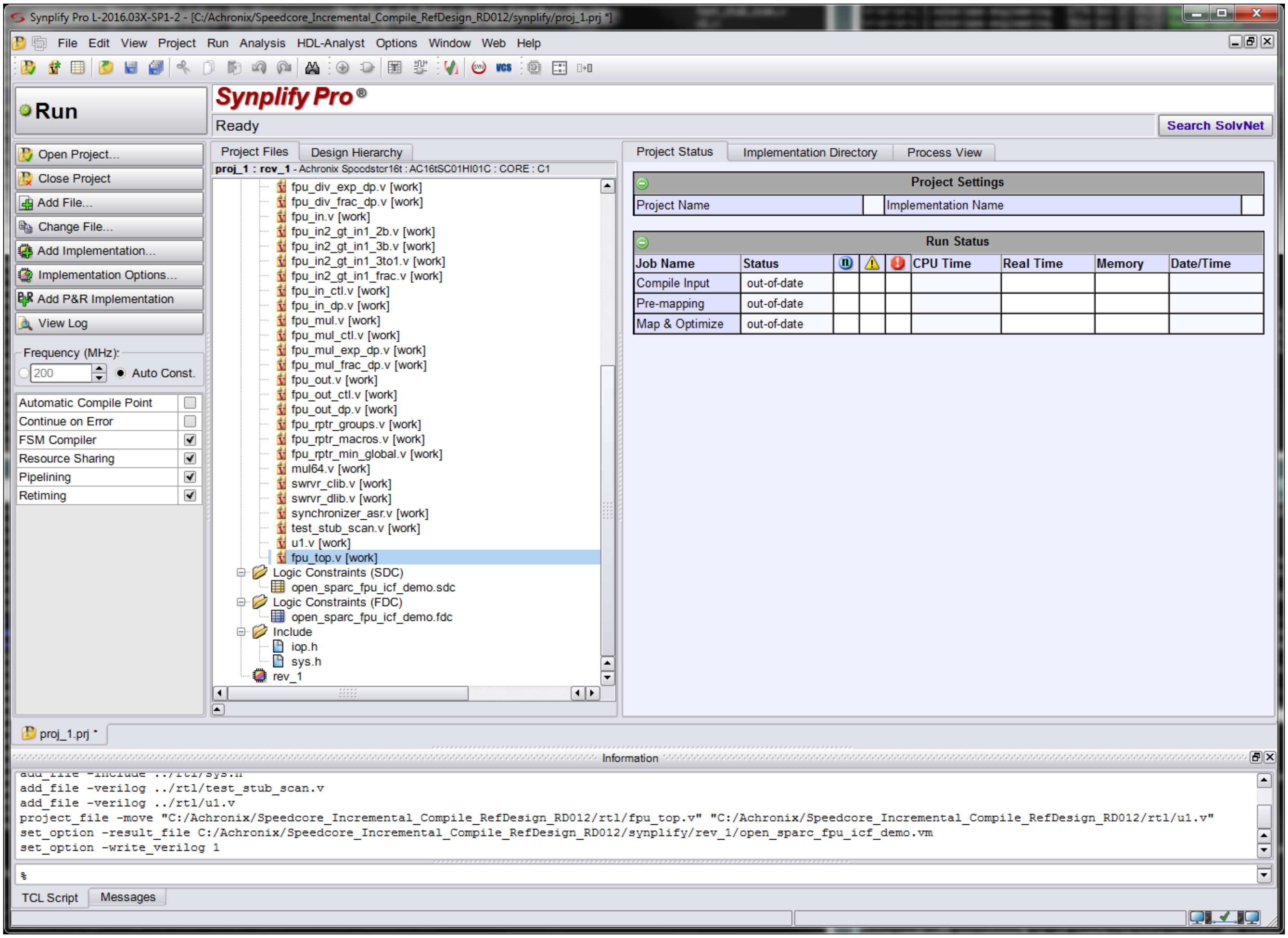


Figure 164: *Synplify Pro Home Screen After File Additions*

Depending on how Synplify Pro is installed, the locations of the Achronix macro libraries may need to be specified. Open the Implementation Options window and then click the Verilog tab (**Project** → **Implementation Options** → **Verilog**). Ensure that `<ACE install location>/libraries` is present in the Include Path Order box. If not, add them by clicking the green **+** and navigating to the `<ACE install location>/libraries` directory. Click **Choose**, then click **OK**.

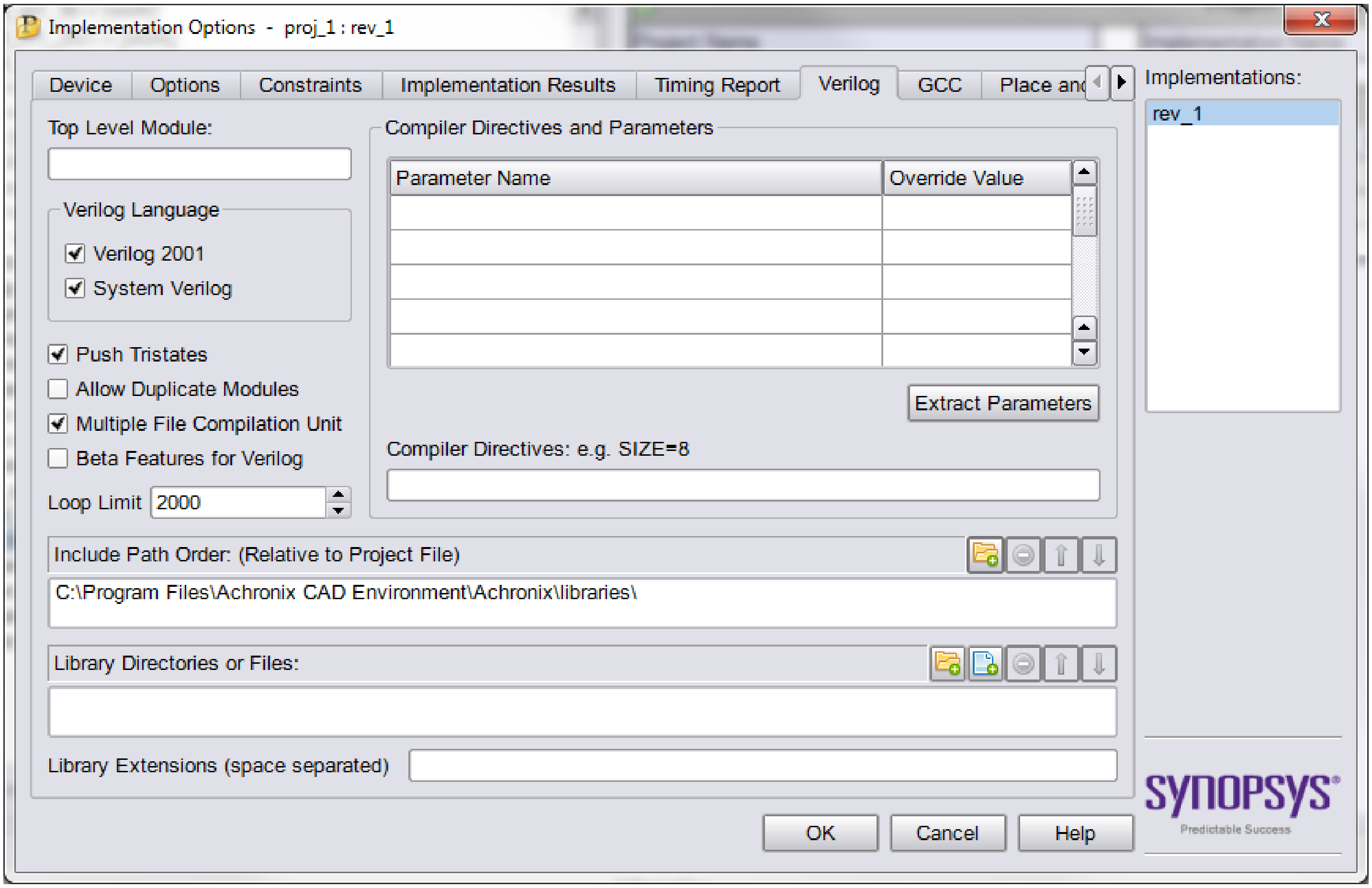


Figure 165: *Implementation Options Window*

Ensure that the Verilog file `<ACE install location>/libraries/device_models/16t_synplify.v` has been added to the project. If it has not, use the **Project** → **Add Source File** dialog box again to add it. Then drag this file to the top of the Verilog files listed to ensure that it is the first one read in.

Finally, select the Achronix technology and part name. Open the Implementation Options Device tab (**Project** → **Implementation Options** → **Device**) and select the **Technology:** and **Part:** name.

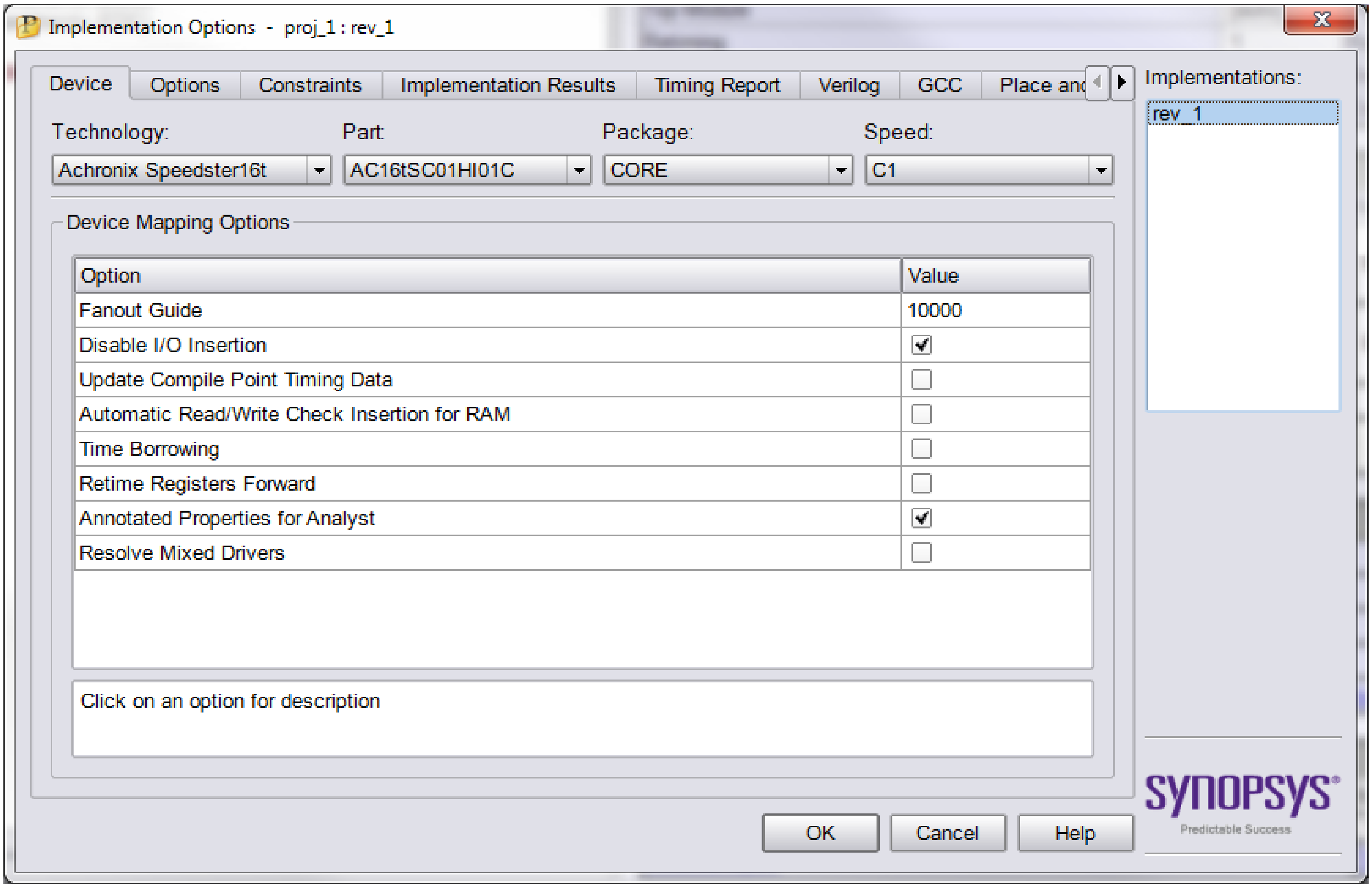


Figure 166: ***Implementation Option Device Tab***

Step 3: Compile the Design in Synplify Pro

Select **Run** → **Compile Only** (or click **F7**) to parse the Verilog and constraints files and enable viewing. If this is the first time the design was compiled and the project has not been saved yet, Synplify Pro may ask to save the project file. Click **Save** to continue.

In order to see the nine compile point constraints defined for this project, open the `constraints/open_sparc_fpu_icf_demo.fdc` file as a text file by navigating to the **Project Files** tab, expand the **Logic Constraints (FDC)** section, and left-mouse click the `open_spacr_fpu_icf_demo.fdc` file, and **Open as Text**. All nine of them are of type locked, as in the example below:

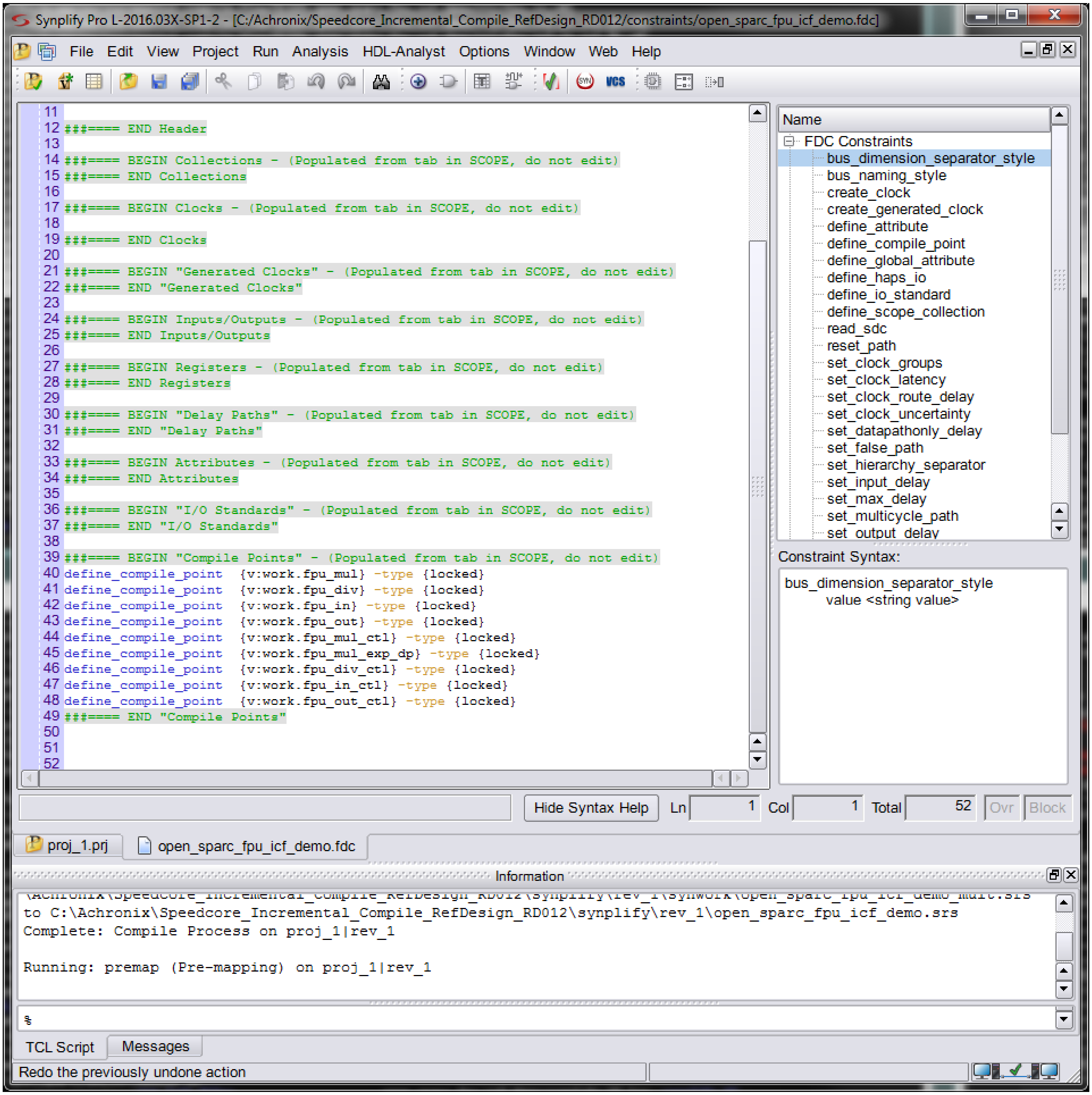


Figure 167: Contents of the *open_sparc_fpu_icf_demo.fdc* File

Each compile point becomes a partition in the ACE tool. If one or more RTL source files are later edited and changed, Synplify Pro and ACE only need to recompile the partitions that have changed, rather than the whole design.

Optionally; instead of adding the `open_sparc_fpu_icf_demo.fdc` file to define the compile points, constraints can also be created or edited using the SCOPE tool. To manually add a new constraint file, click the ( ) **new constraint file** button, and then click the **Compile Points** tab. To open the existing `open_sparc_fpu_icf_demo.fdc` file in the SCOPE tool, double-click the file name in the **Project Files** tab, and then click the **Compile Points** tab. Then, to add a new Compile Point, select the first blank row in the table, double-click in the View field to bring up a drop-down list of available view names, and select the one desired. Then double-click in the Type field to set the compile-point type. ACE treats all compile points as locked for purposes of placement and routing, but soft or hard compile points can be used in synthesis if locked results in poor QoR.

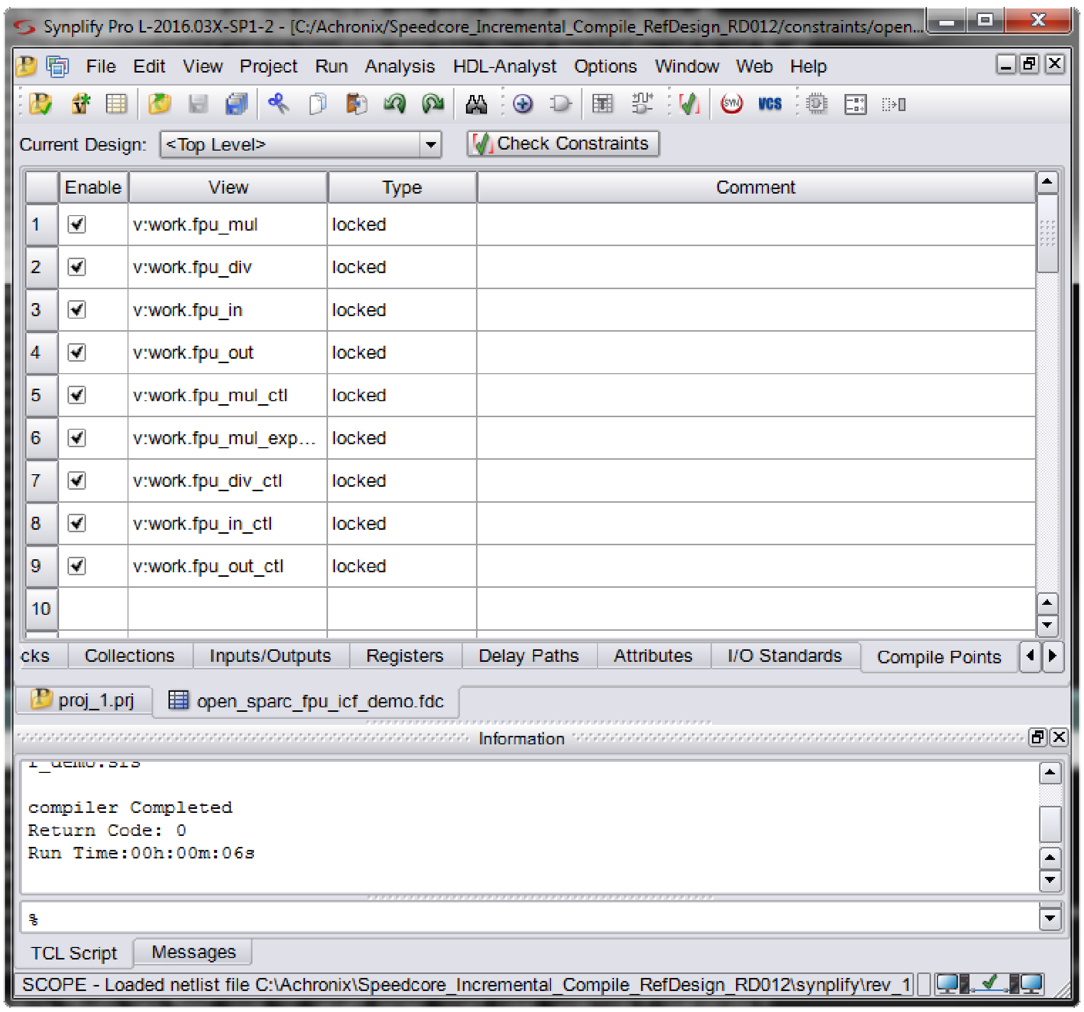


Figure 168: ***Compile Points Tab Within the Synplify Pro SCOPE View***

Close the SCOPE View window and do not save any changes.

> **Note**
>
> Synplify-Pro can be configured to create the compile points automatically. To experiment with this option, open the Implementation Options window, select the Options tab, and check the **Auto Compile Point** option. This option uses various heuristics (such as the sizes of the modules, the number of pins and the presence of timing constraints) to select a set of module views as compile points. These may be in addition to any compile points manually specified as constraints.

For this tutorial ensure that the **Auto Compile Point** option is un-checked, then click **OK** to close the Implementation Options window.

Lastly, click **Run** to complete the mapping of the design.

For more information see Chapter 11, "Working with Compile Points" in the document *Synopsys FPGA Synthesis Synplify Pro for Achronix User Guide*, located in the Synplify Pro installation directory under `/doc`.

> **Caution!**
>
> Windows users may encounter an error with `m_generic.exe` while using compile points. This condition is caused by an issue with parallel synthesis jobs in the current version of Synplify Pro for Achronix. This situation is being addressed by Synopsys and is expected to be rectified in an upcoming release. If Synplify encounters this error, select **Options** → **Configure Compile Point Process**, change the "4" in the box to "1" as shown below.

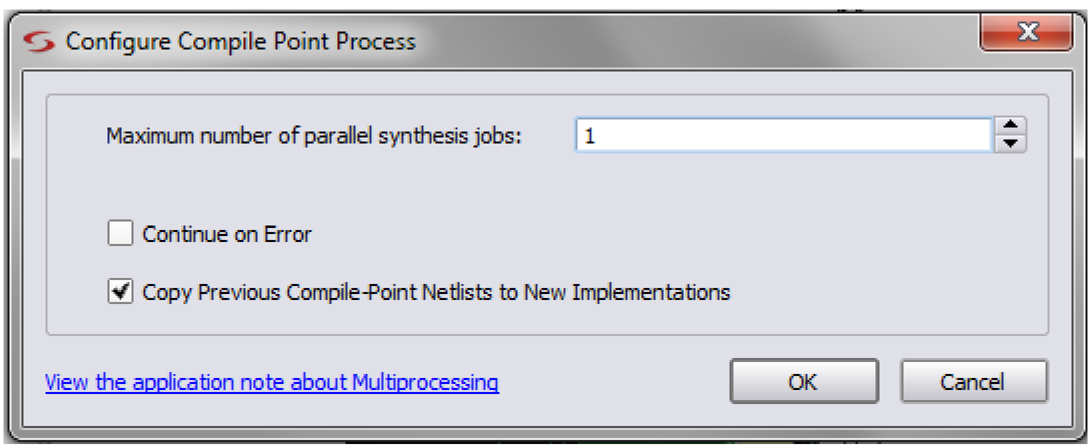


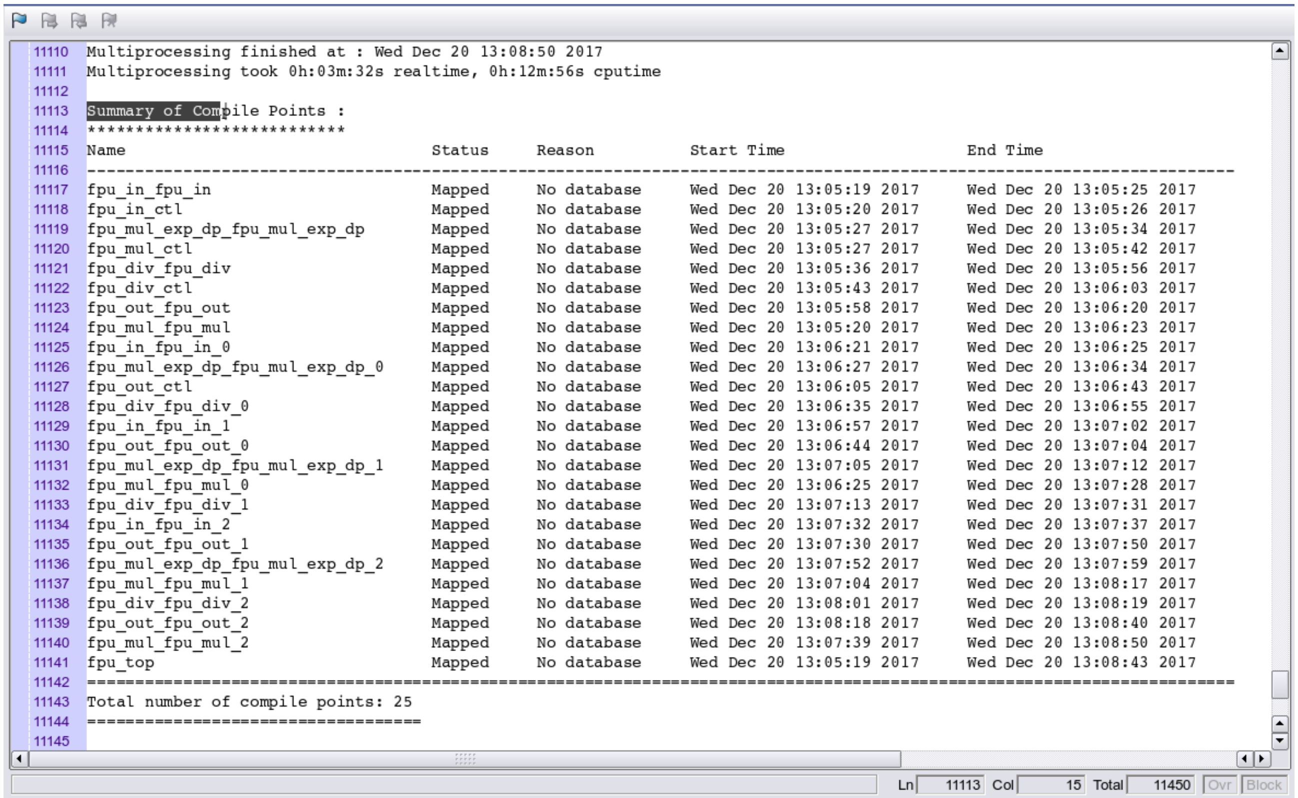
Figure 169: *Configure Compile Point Process Dialog Box*

Step 4: Review Synplify Results

This step reviews some of the files and features available to better understand the behavior of Synplify Pro with compile-point constraints.

Synplify-Pro Log File

Using either the Synplify Pro GUI or another text editor, open the Synplify Pro log file `Speedcore_Incremental_Compile_RefDesign_RD012/synplify/rev_1/open_sparc_fpu_icf_demo.srr` and search for the section titled "Summary of Compile Points".



```
Multiprocessing finished at : Wed Dec 20 13:08:50 2017
Multiprocessing took 0h:03m:32s realtime, 0h:12m:56s cputime

Summary of Compile Points :
***************************
Name                                Status     Reason          Start Time                    End Time
------------------------------------------------------------------------------------------------------------------------
fpu_in_fpu_in                       Mapped     No database     Wed Dec 20 13:05:19 2017      Wed Dec 20 13:05:25 2017
fpu_in_ctl                          Mapped     No database     Wed Dec 20 13:05:20 2017      Wed Dec 20 13:05:26 2017
fpu_mul_exp_dp_fpu_mul_exp_dp       Mapped     No database     Wed Dec 20 13:05:27 2017      Wed Dec 20 13:05:34 2017
fpu_mul_ctl                         Mapped     No database     Wed Dec 20 13:05:27 2017      Wed Dec 20 13:05:42 2017
fpu_div_fpu_div                     Mapped     No database     Wed Dec 20 13:05:36 2017      Wed Dec 20 13:05:56 2017
fpu_div_ctl                         Mapped     No database     Wed Dec 20 13:05:43 2017      Wed Dec 20 13:06:03 2017
fpu_out_fpu_out                     Mapped     No database     Wed Dec 20 13:05:58 2017      Wed Dec 20 13:06:20 2017
fpu_mul_fpu_mul                     Mapped     No database     Wed Dec 20 13:05:20 2017      Wed Dec 20 13:06:23 2017
fpu_in_fpu_in_0                     Mapped     No database     Wed Dec 20 13:06:21 2017      Wed Dec 20 13:06:25 2017
fpu_mul_exp_dp_fpu_mul_exp_dp_0     Mapped     No database     Wed Dec 20 13:06:27 2017      Wed Dec 20 13:06:34 2017
fpu_out_ctl                         Mapped     No database     Wed Dec 20 13:06:05 2017      Wed Dec 20 13:06:43 2017
fpu_div_fpu_div_0                   Mapped     No database     Wed Dec 20 13:06:35 2017      Wed Dec 20 13:06:55 2017
fpu_in_fpu_in_1                     Mapped     No database     Wed Dec 20 13:06:57 2017      Wed Dec 20 13:07:02 2017
fpu_out_fpu_out_0                   Mapped     No database     Wed Dec 20 13:06:44 2017      Wed Dec 20 13:07:04 2017
fpu_mul_exp_dp_fpu_mul_exp_dp_1     Mapped     No database     Wed Dec 20 13:07:05 2017      Wed Dec 20 13:07:12 2017
fpu_mul_fpu_mul_0                   Mapped     No database     Wed Dec 20 13:06:25 2017      Wed Dec 20 13:07:28 2017
fpu_div_fpu_div_1                   Mapped     No database     Wed Dec 20 13:07:13 2017      Wed Dec 20 13:07:31 2017
fpu_in_fpu_in_2                     Mapped     No database     Wed Dec 20 13:07:32 2017      Wed Dec 20 13:07:37 2017
fpu_out_fpu_out_1                   Mapped     No database     Wed Dec 20 13:07:30 2017      Wed Dec 20 13:07:50 2017
fpu_mul_exp_dp_fpu_mul_exp_dp_2     Mapped     No database     Wed Dec 20 13:07:52 2017      Wed Dec 20 13:07:59 2017
fpu_mul_fpu_mul_1                   Mapped     No database     Wed Dec 20 13:07:04 2017      Wed Dec 20 13:08:17 2017
fpu_div_fpu_div_2                   Mapped     No database     Wed Dec 20 13:08:01 2017      Wed Dec 20 13:08:19 2017
fpu_out_fpu_out_2                   Mapped     No database     Wed Dec 20 13:08:18 2017      Wed Dec 20 13:08:40 2017
fpu_mul_fpu_mul_2                   Mapped     No database     Wed Dec 20 13:07:39 2017      Wed Dec 20 13:08:50 2017
fpu_top                             Mapped     No database     Wed Dec 20 13:05:19 2017      Wed Dec 20 13:08:43 2017
========================================================================================================================
Total number of compile points: 25
==================================
```


Figure 170: *Synplify-Pro Log File Showing Summary of Compile Points Section*

The log file `rev_1/synlog/open_sparc_fpu_icf_demo_fpga_mapper.srr` lists a summary line for each of the defined compile points. Each compile point is an instance of the modules defined in the FDC file. All of these compile points are marked as **Mapped** (in this case "No database" because the design is being mapped for the first time). The timestamp of the last compile for each indicate they are all mapped at about the same time. Immediately below that section is a reference to a separate `.srr` log file file for each compile point.

> **Note**
>
> The "**Summary of Compile Points**" section may contain different **Name** entries than those that were defined in the FDC file. These can be instances of those modules.

The log file may contain the following warnings:

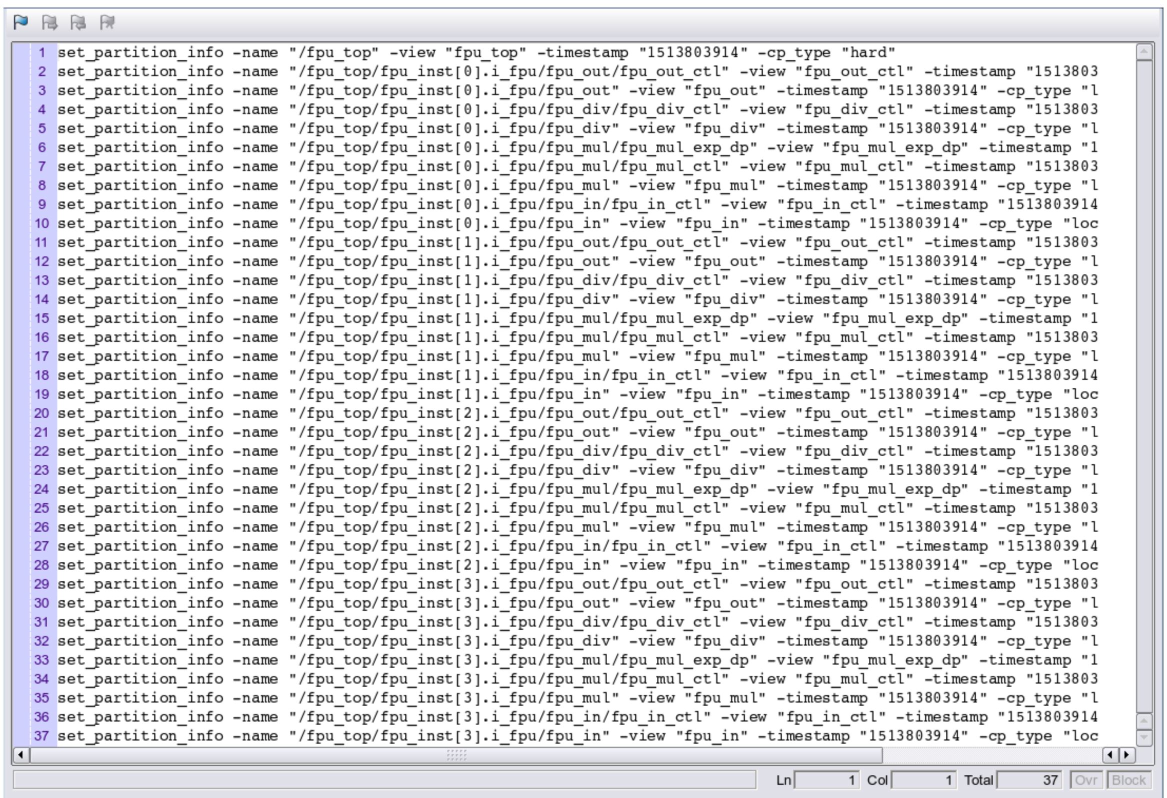
```
@N: MF104 :|Found compile point of type locked on View view:work.fpu_in_ctl(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_in(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul_ctl(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul_exp_dp(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_div_ctl(verilog)
```

```
@N: MF104 :|Found compile point of type locked on View view:work.fpu_div(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_out_ctl(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_out(verilog)
```

These warnings are due to a a caveat when using attributes with compile points. Attributes can be used when setting constraints for compile points. However, when using `syn_hier` on a compile point, the only valid value is *flatten*. All other values of this attribute (e.g., hard) are ignored for compile points. The `syn_hier` attribute behaves normally for all other module boundaries not defined as compile points.

ACE Partitioning Constraints File

The file `synplyfy/rev_1/open_sparc_fpu_icf_demo_partition.prt` is written by Synplify for inclusion in the ACE project. This file contains TCL commands that define the Synplify-Pro compile points as partitions in ACE. Each command contains both the instance and view names of each partition, as well as its timestamp and compile-point type. There are many more partitions (37) listed in the `.prt` file than there were compile points listed in the Synplify log file because the partitions represent instances, while the compile points represent modules (many of the modules are instantiated multiple times in this design). The number of these instances match the number of the compile points found in the "Summary of Compile Points" section.



```
set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1513803914" -cp_type "hard"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
```


Figure 171: *Contents of the open_sparc_fpu_icf_demo_partition.prt File*

Technology View

Click the ( ) **Technology View** button to open the design schematic, then select one of the fpu_inst instances. These instances can be identified by expanding the Instances/Groups folder and then left-mouse click one to select it. The selected instance is highlighted with a red boundary in the Tech popup view.

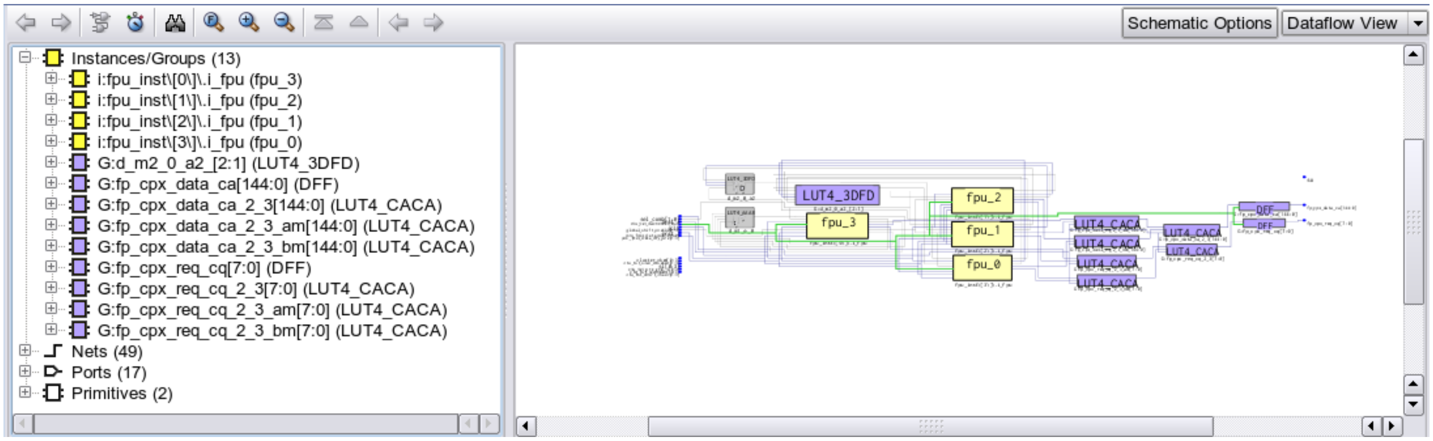


Figure 172: ***Synplify Pro Initial Technology View Top-level Schematic***

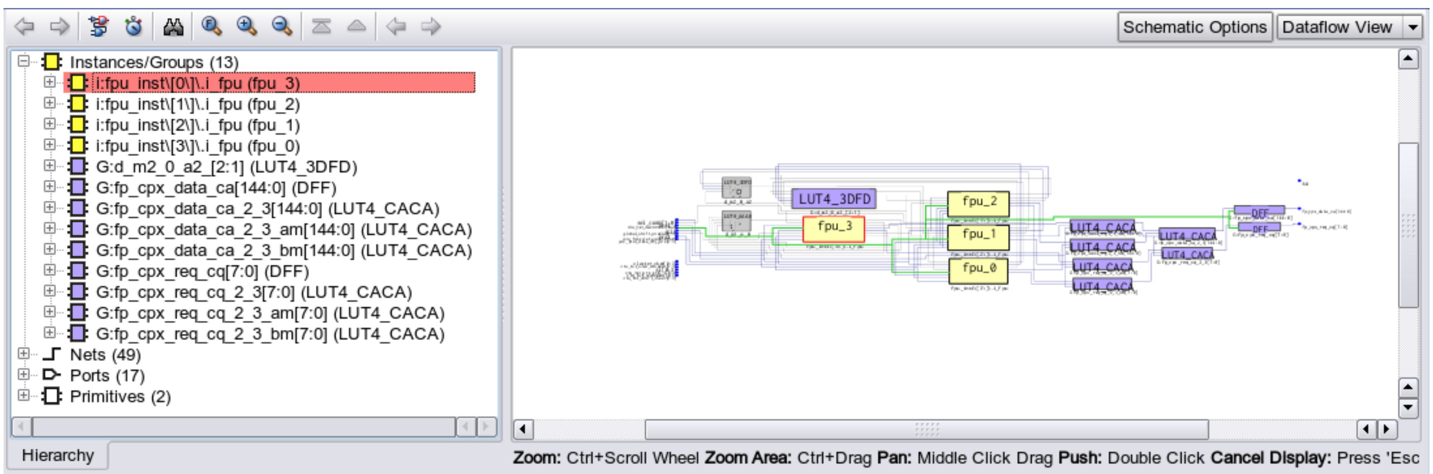


Figure 173: ***?***

Use the right mouse button to push into that level of the hierarchy. The schematic then updates. The locked and hard partitions have a green background color while the default instance background color is yellow. In the schematic area, use the right now to either push or pop hierarchy levels, depending on where the mouse is located when clicked.

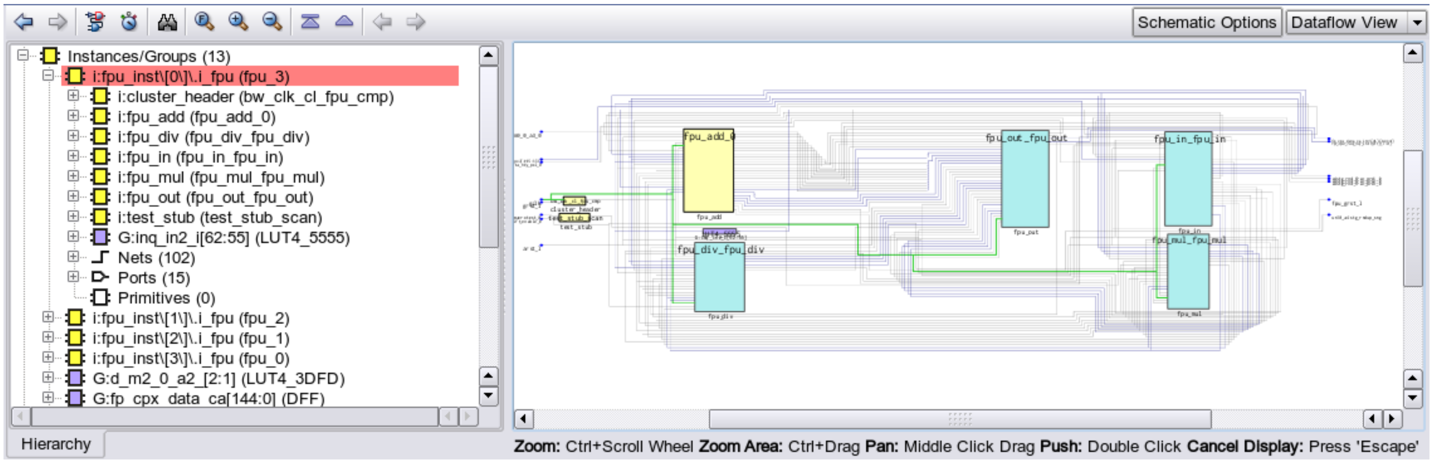


Figure 174: ***Synplify Pro Technology View***

Exit from Synplify Pro. Next, the design must be placed and routed in ACE.

Step 5: Set up the ACE Project

Start the ACE GUI. Under Linux, execute:

```
% cd <your work area>/Speedcore_Incremental_Compile_RefDesign_RD012/ace
% ace
```

Under Windows, double click the ACE icon.

Then create a new project with **File** → **Create Project**. Click **Browse** to navigate through the filesystem to ensure that the project is created under the subdirectory `Speedcore_Incremental_Compile_RefDesign_RD012/ace` and click **OK**. Use `proj_1` for the project name, and `impl_1` for the implementation name. Click **Finish**.

The ACE home screen appears with an empty project named proj_1 and an implementation named impl_1, as in the following screen shot:

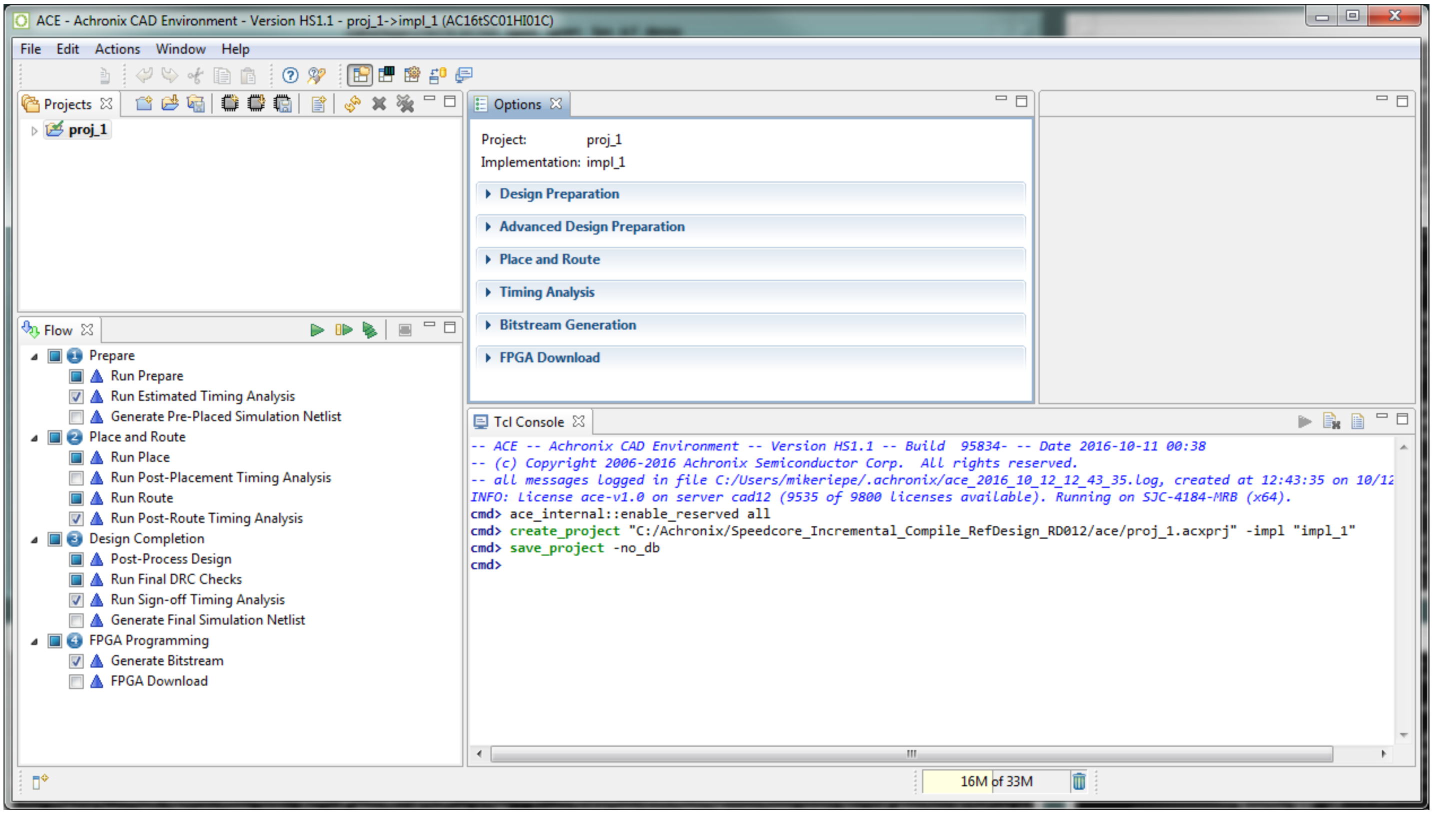


Figure 175: *ACE Home Screen*

Select **File** → **Add Project Source Files…**, then click **Speedcore_Incremental_Compile_RefDesign_RD012** in the pathname bar to locate the source files and open the following dialog box:

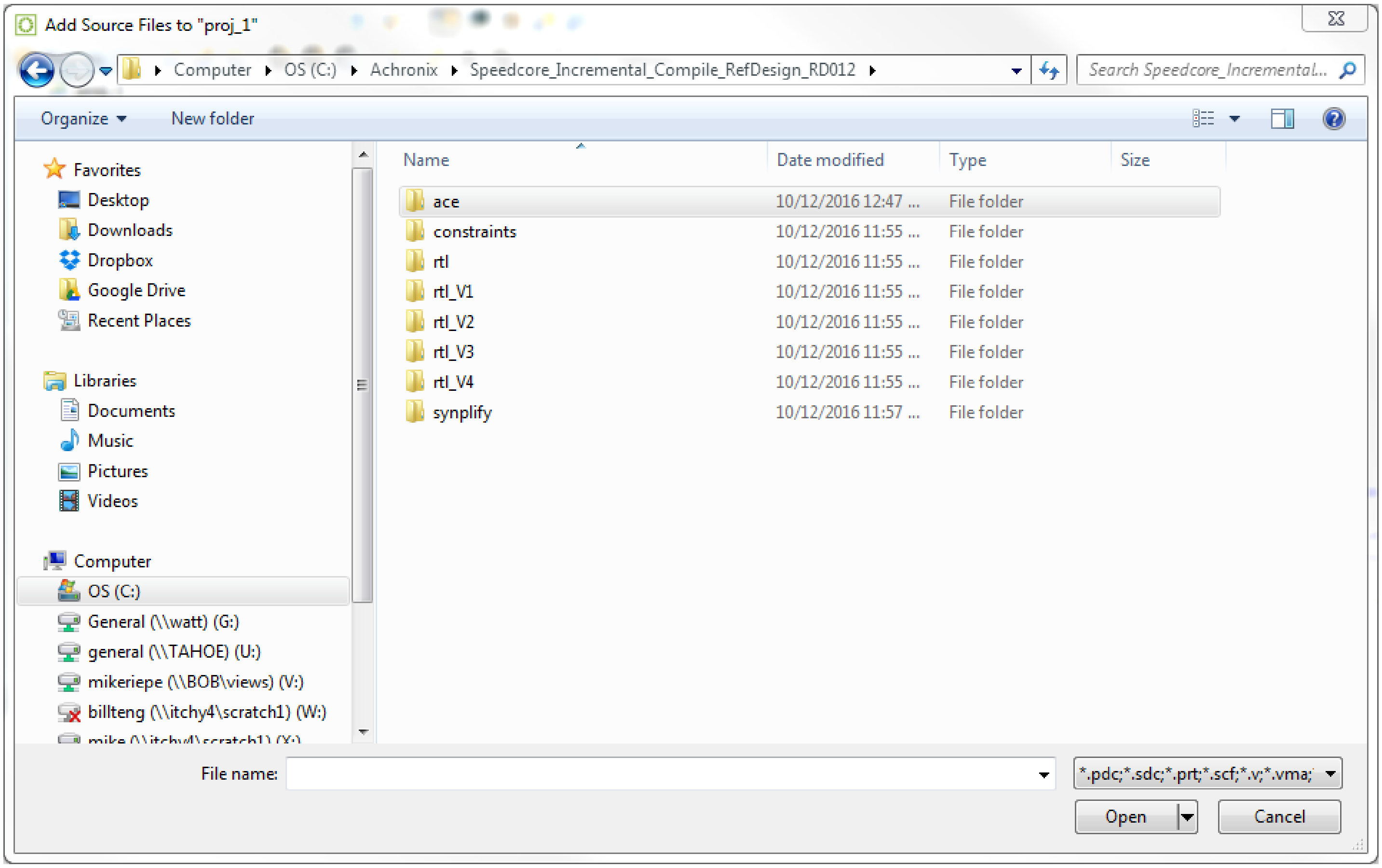


Figure 176: ***Design Source Files Dialog***

Navigate to the constraints directory, select the `open_sparc_fpu_icf_demo.sdc` file, and then click **OK**. Bring up the Add Project Source Files dialog box again, navigate to the directory `synplify/rev_1/`, control-click to select the files `open_sparc_fpu_icf_demo.vm` and `open_sparc_fpu_icf_demo_partition.prt`, and click **OK** to add them to the project.

Finally, in the Options tab under "Design Preparation", verify that the **Target Device** is the same device name used in Synplify, and verify that the **Enable Incremental Compile** implementation option checkbox is checked. All of the files just added appear under the `ace/Netlists` and `ace/Constraints` folders of the Projects tab.

Tip

In the **Projects Perspective** → **Projects View** click the triangle next to Constraints to list out the constraint files, etc.

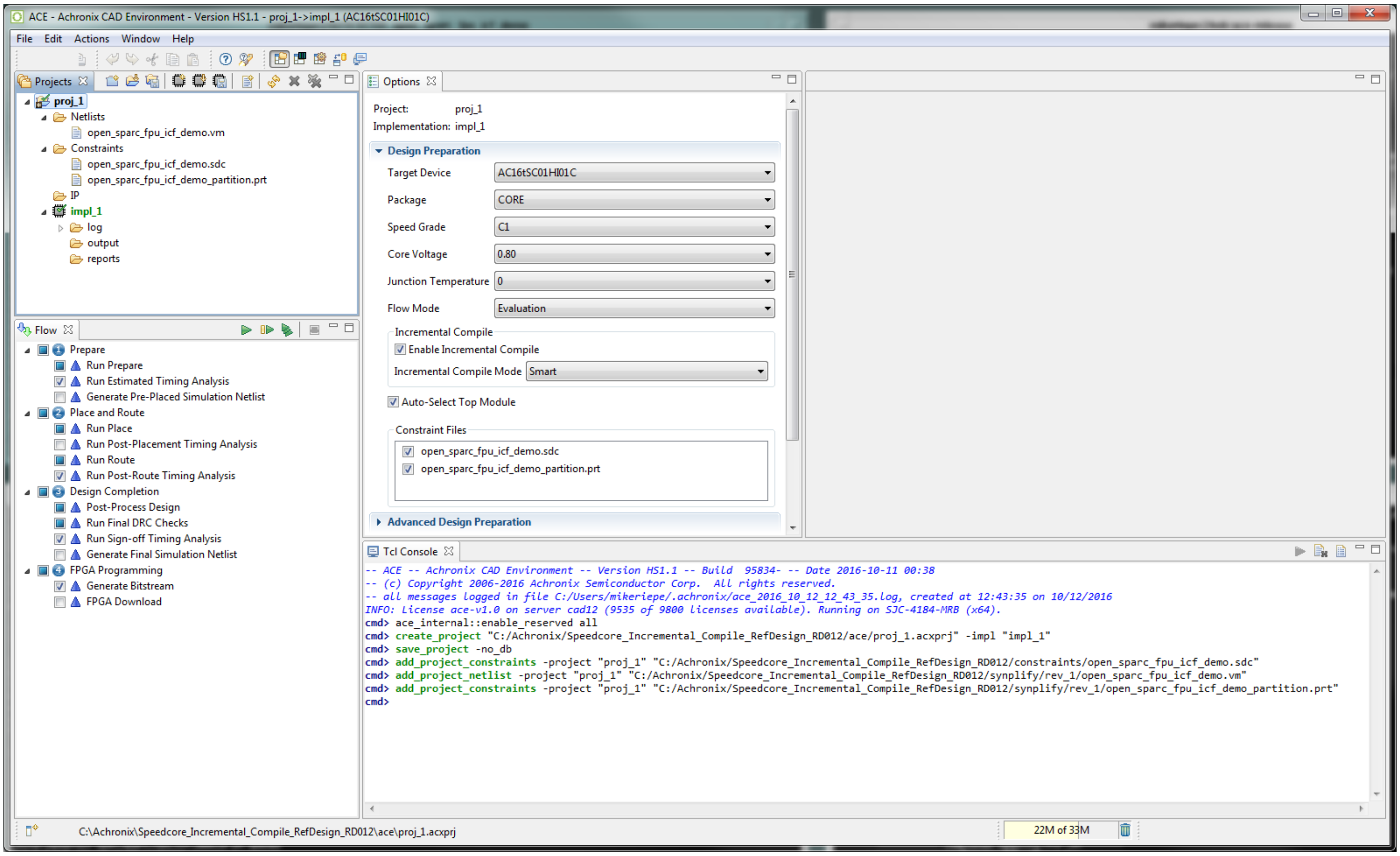


Figure 177: *ACE Projects Tab*

Recall from Step 4 (see page 376) that the `open_sparc_fpu_icf_demo_partition.prt` file (added to the project above) contains the partition definitions exported from Synplify Pro. The presence of this constraint file and the Enable Incremental Compile implementation option are the only configuration changes that distinguish the incremental compile flow from the standard non-incremental flow.

Immediately under the **Enable Incremental Compile** implementation option checkbox is a drop-down box for the **Incremental Compile Mode** implementation option. Available values are strict and smart. Strict mode ensures that placement of locked instances in unchanged partitions are completely preserved. Smart mode (the default) allows ACE to try to intelligently preserve placement in locked partitions for better design performance.

Step 6: Compile the Design in ACE

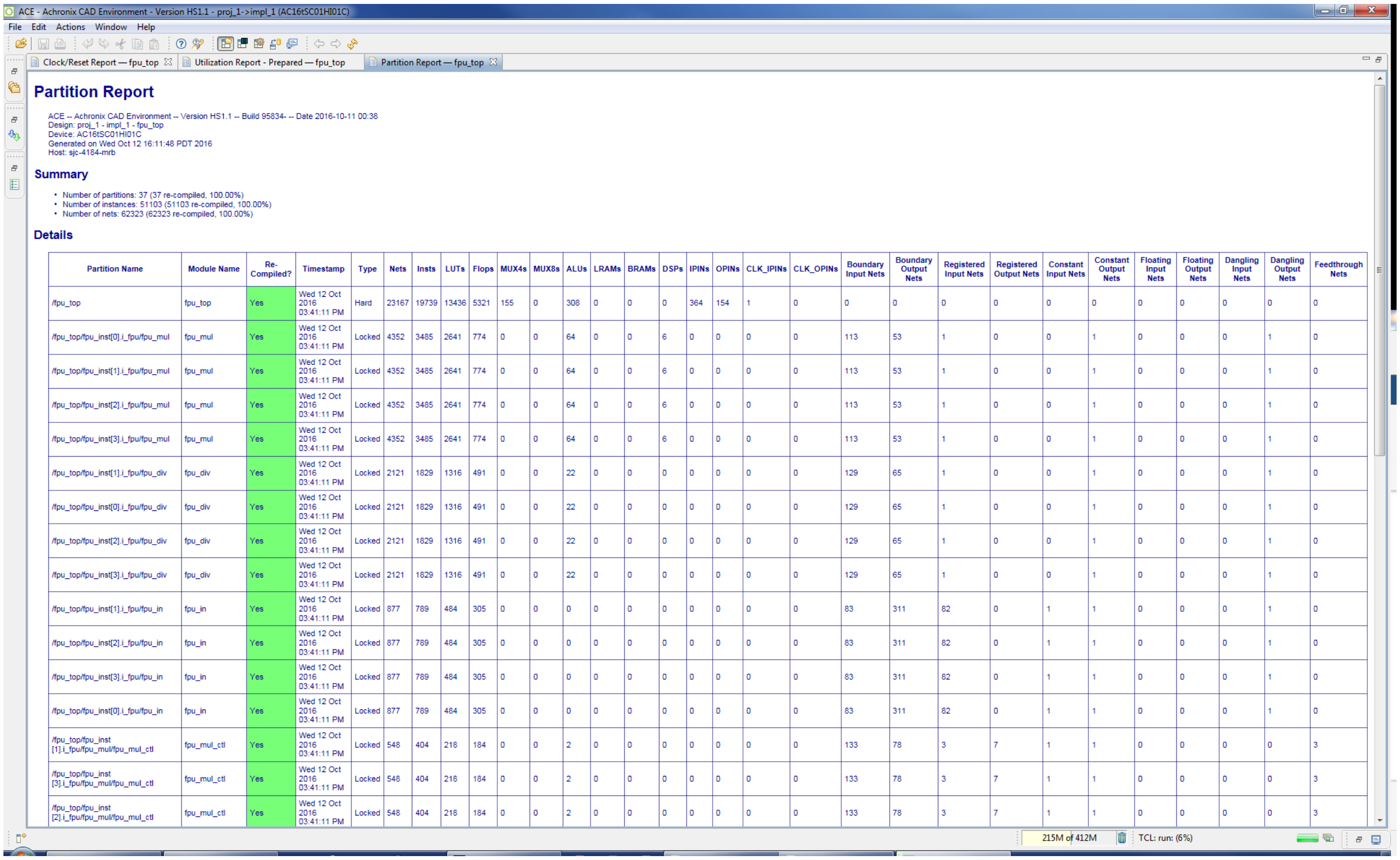
In the Flow tab, uncheck the **Run Sign-off Timing Analysis** and **Generate Bitstream** flow step checkboxes to save some runtime. Then click the green triangle ( ) in the upper-right corner of the Flow view to run the prepare, placement, and routing flow. When the ACE flow completes, a green check mark appears by the Run Final DRC Checks flow step in the Flow view.

Step 7: Review ACE Results

Next is a review of some of the features available to help in understanding and optimizing the partition constraints.

Partition Report

While the ACE flow is running, the Partition Report can be viewed at any time after the completion of the Run Prepare flow step. In the ACE GUI, the report opens automatically in the Editor Area of the Project perspective.



Partition Report

ACE -- Achronix CAD Environment -- Version HS1.1 -- Build 95834- -- Date 2016-10-11 00:38
Design: proj_1 - impl_1 - fpu_top
Device: AC16tSC01HI01C
Generated on Wed Oct 12 16:11:48 PDT 2016
Host: sjc-4184-mrb

Summary

- Number of partitions: 37 (37 re-compiled, 100.00%)
- Number of instances: 51103 (51103 re-compiled, 100.00%)
- Number of nets: 62323 (62323 re-compiled, 100.00%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs
/fpu_top	fpu_top	Yes	Wed 12 Oct 2016 03:41:11 PM	Hard	23167	19739	13436	5321	155	0	308	0	0	0	364	154	1	0
/fpu_top/fpu_inst[0].i_fpu/fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0
/fpu_top/fpu_inst[1].i_fpu/fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0
/fpu_top/fpu_inst[2].i_fpu/fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0
/fpu_top/fpu_inst[3].i_fpu/fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0
/fpu_top/fpu_inst[1].i_fpu/fpu_div	fpu_div	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	2121	1829	1316	491	0	0	22	0	0	0	0	0	0	0
/fpu_top/fpu_inst[0].i_fpu/fpu_div	fpu_div	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	2121	1829	1316	491	0	0	22	0	0	0	0	0	0	0
/fpu_top/fpu_inst[2].i_fpu/fpu_div	fpu_div	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	2121	1829	1316	491	0	0	22	0	0	0	0	0	0	0
/fpu_top/fpu_inst[3].i_fpu/fpu_div	fpu_div	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	2121	1829	1316	491	0	0	22	0	0	0	0	0	0	0
/fpu_top/fpu_inst[1].i_fpu/fpu_in	fpu_in	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0
/fpu_top/fpu_inst[2].i_fpu/fpu_in	fpu_in	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0
/fpu_top/fpu_inst[3].i_fpu/fpu_in	fpu_in	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0
/fpu_top/fpu_inst[0].i_fpu/fpu_in	fpu_in	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0
/fpu_top/fpu_inst [1].i_fpu/fpu_mul/fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	548	404	218	184	0	0	2	0	0	0	0	0	0	0
/fpu_top/fpu_inst [3].i_fpu/fpu_mul/fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	548	404	218	184	0	0	2	0	0	0	0	0	0	0
/fpu_top/fpu_inst [2].i_fpu/fpu_mul/fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 03:41:11 PM	Locked	548	404	218	184	0	0	2	0	0	0	0	0	0	0

Partition Name	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthrough Nets
/fpu_top	0	0	0	0	0	0	0	0	0	0	0
/fpu_top/fpu_inst[0].i_fpu/fpu_mul	113	53	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[1].i_fpu/fpu_mul	113	53	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[2].i_fpu/fpu_mul	113	53	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[3].i_fpu/fpu_mul	113	53	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[1].i_fpu/fpu_div	129	65	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[0].i_fpu/fpu_div	129	65	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[2].i_fpu/fpu_div	129	65	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[3].i_fpu/fpu_div	129	65	1	0	0	1	0	0	0	1	0
/fpu_top/fpu_inst[1].i_fpu/fpu_in	83	311	82	0	1	1	0	0	0	1	0
/fpu_top/fpu_inst[2].i_fpu/fpu_in	83	311	82	0	1	1	0	0	0	1	0
/fpu_top/fpu_inst[3].i_fpu/fpu_in	83	311	82	0	1	1	0	0	0	1	0
/fpu_top/fpu_inst[0].i_fpu/fpu_in	83	311	82	0	1	1	0	0	0	1	0
/fpu_top/fpu_inst [1].i_fpu/fpu_mul/fpu_mul_ctl	133	78	3	7	1	1	0	0	0	0	3
/fpu_top/fpu_inst [3].i_fpu/fpu_mul/fpu_mul_ctl	133	78	3	7	1	1	0	0	0	0	3
/fpu_top/fpu_inst [2].i_fpu/fpu_mul/fpu_mul_ctl	133	78	3	7	1	1	0	0	0	0	3

Figure 178: ***ACE Partition Report After First Incremental Compile Iteration***

First look at the Summary section. The report shows the total number of partitions and the number that were recompiled (in this case 100% because this was the first pass through the flow). Also listed are the number of instances and nets that are owned by a partition, plus the number that were recompiled by ACE (again, 100% of each). Placement runtimes are proportional to the number of recompiled instances, and routing runtimes are proportional to the number of recompiled nets.

The Details section displays a table with one row for each partition. Columns are printed as follows:

- Partition Name (which are the same as the instance name in the unflattened RTL)
- Module Name (derived from the module name in the RTL)
- Re-Compiled? column ("yes" or "no")
- Timestamp (time and date when it was last compiled in Synplify Pro)
- Type (only "Hard" and "Locked" are supported by ACE)
- The number of nets and instances owned by the partition

- A series of columns with instance counts for LUTs, Flops (DFFs), ALUs, LRAMs, BRAMs, etc.

The final eleven columns in the Details section provide information about the boundary nets of each partition.This information is useful in analyzing the suitability of each module for partitioning and to suggest ways in which the design may be improved to make it more amenable to partitioning. These include:

- Two columns counting the number of input nets and output nets crossing the boundary. These correspond to the ports of the original RTL module that have been flattened away. An input net has its driver outside the partition, while an output net has its driver inside the partition. The larger the ratio of boundary nets to instances in a partition, the more likely it is that the placement and routing of the partition is disturbed when neighboring partitions are recompiled (this is a corollary of Rent's Rule).
- Two columns with the number of input and output boundary nets that are registered. Registering boundary ports is always a good idea as it can be harder to maintain timing closure of cross-boundary paths when a partition or its parent needs to be recompiled.
- Two columns with the number of input and output boundary nets driven by a constant. Designers often tie-off unused inputs or outputs of a block and assume that those constants are optimized away by the logic synthesis tools. However, logic synthesis must assume that those constants may change in the future, so constant propagation cannot be performed across locked partition boundaries. Locking can result in a netlist that is much larger than expected. It is better to define a compile point on an RTL wrapper module that encloses input pin constants inside the partition and output-pin constants outside the partition. This method provides logic synthesis with the freedom to eliminate gates made redundant by the constants.
- Four columns with the number of input and output boundary nets that are floating and dangling, respectively. Floating nets have input pin loads with no driver, and dangling nets have a driver with no input pin loads. Similarly to constant boundary nets, logic synthesis is not able to optimize away floating and dangling logic across locked partition boundaries. Again, if a design has pins on a module that can logically be left unconnected, it is usually best to create a wrapper module so that unconnected inputs are enclosed within the partition and unconnected outputs are outside the partition, and then define the wrapper module as the partition instead.
- One column with the number of feedthrough nets. A feedthrough net enters an input pin of a module and exits through an output pin without driving any logic inside the partition. Again because logic synthesis cannot optimize logic across Locked partition boundaries, and it cannot eliminate pins on either Locked or Hard partition boundaries, it is best to eliminate feedthrough nets from the design. Feedthroughs impose constraints on synthesis, placement, and routing that can results in unnecessary delay and routing congestion.

> **Note**
>
> The table is sorted so that partitions with a recompiled state of "Yes" appear first, then sorted by number of instances.

Floorplan View

After the Routing flow step has completed, switch to the Floorplanner perspective to view the results. In the Floorplanner View (see page 53) flyout palette, under the Layers section, turn on visibility for Instances, but turn off visibility for Sites, Clock Routes, and Non-clock Routes.

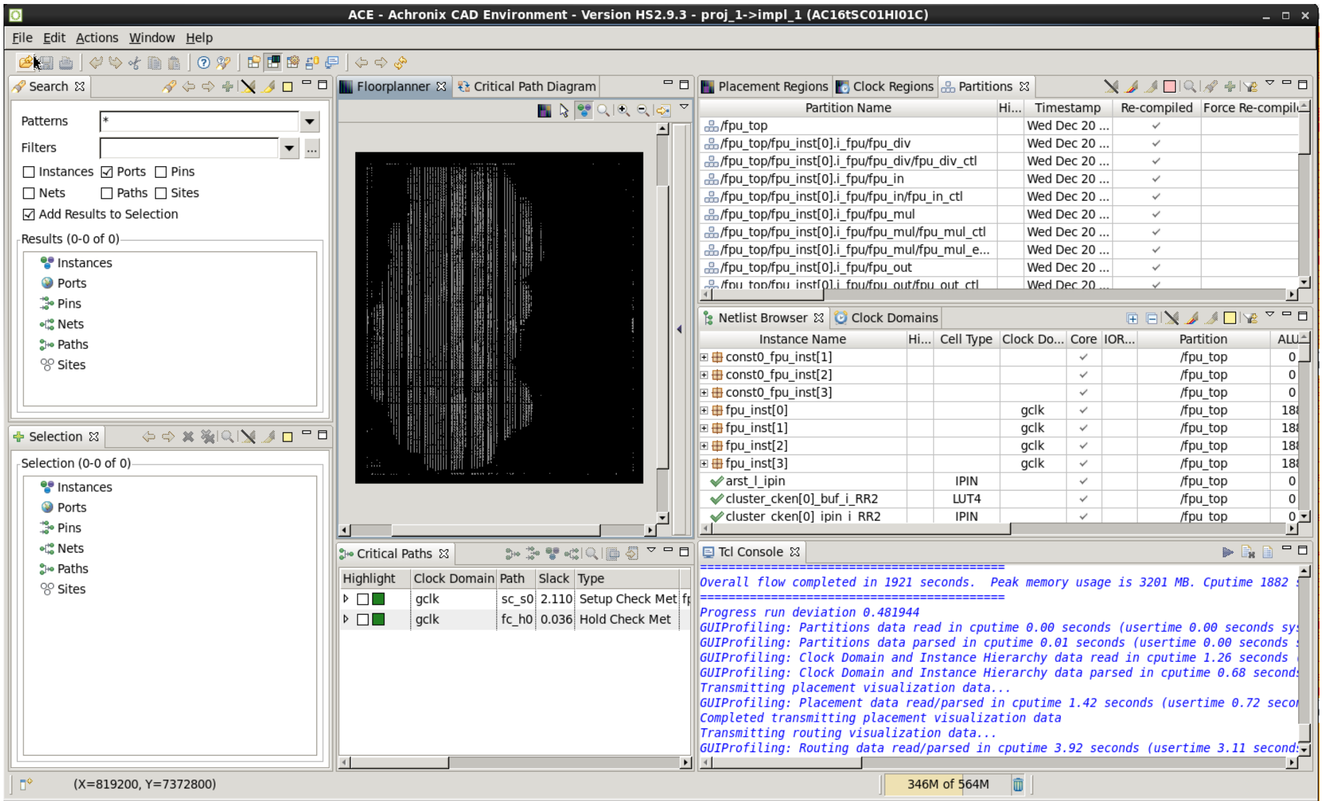


Figure 179: *ACE Floorplanner Perspective After First Incremental Compile Iteration*

Switch to the Partitions View (see page 117) to see a table with one row for each partition name. Columns exist for the Partition Timestamp; Re-Compiled status (a check-mark or not); the number of Flops, LUTs, ALUs, BRAMs, LRAMs, and Others (IOs, sources, etc); and the number of Cumulative Flops, LUTs, ALUs, BRAMs, LRAMs, and Others. The number of instances of each type includes only those instances directly owned by the given partition. The number of cumulative instances of each type includes instances owned by the given partition, as well as all child partitions below that partition in the RTL hierarchy.

The column named Force Recompile on Next Run provides a mechanism to override the partition timestamp during the *next* pass through ACE. Right-click anywhere in the row for a partition and select **Force Partition Changed**. A check mark appears in the Force Recompile on Next Run column, and the partition is re-placed and re-routed the next time the flow is run during this ACE session, even if there were no RTL changes nor recompilation in Synplify-Pro. Right-click again and select **Un-Force Partition Changed** to remove the check mark, or else the checkmark is removed automatically when the flow is run later in this ACE session.

The column named Highlight Color displays a box with the partition highlight color. It should currently be empty. Right-click the row for a partition and select Highlight Partition to highlight the partition instances in the Floorplanner view with the current highlight color for the Partition tab. In the toolbar above the table, click the ( ) Choose Highlight Color tool to change the highlight color to be used in the next Highlight command. Right-click the partition row and select **Un-Highlight Partition** to disable highlighting of the partition instances. Alternatively, the ( ) **Highlight Partition** and ( ) **Un-Highlight Partition** tools in the toolbar can be used to highlight the currently selected partition in the table.

Finally, ensure that none of the partitions are highlighted, and click the ( ) Auto-Highlight Partitions tool in the toolbar to highlight all of the partitions with an automatically selected color:

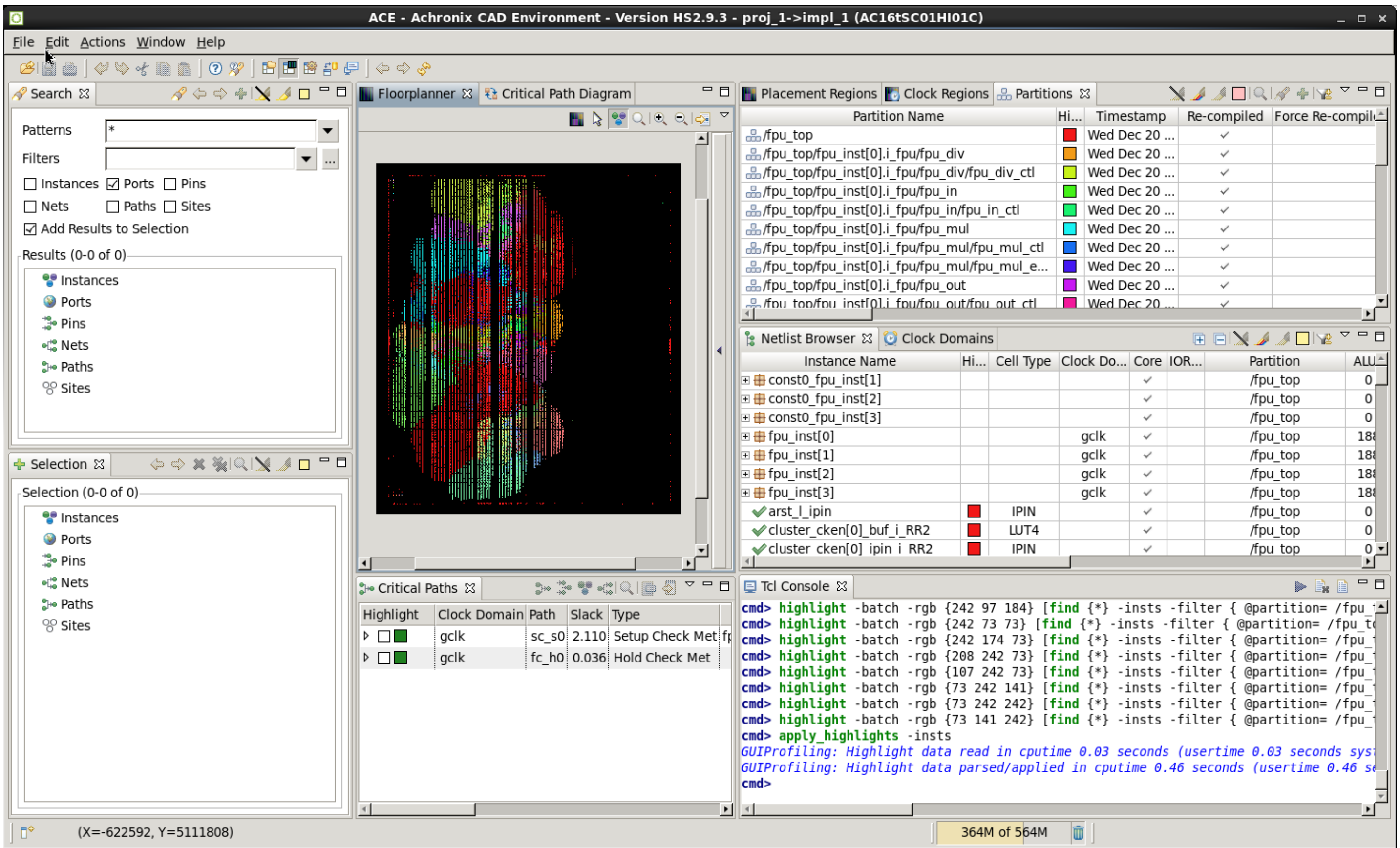


Figure 180: *Highlighting Partitions in ACE After First Incremental Compile Iteration*

The ( ) auto-highlight feature is an extremely powerful tool in understanding the logical and physical connectivity relationships between partitions. Generally, instances in a partition are placed in close proximity to each other if the connectivity within the partition is stronger than the connectivity outside (i.e., Rent's Rule: the number of ports on the partition being much smaller than the number of instances). The instances in one partition are generally placed close to the instances of other partitions with strong connectivity between them, and farther away from other partitions with weak connectivity. Specifically, if a partition has instances scattered over a wide area, it means that the instances are more strongly connected to other partitions than they are to each other. It may be better to remove that partition from the partition constraints. Placement and routing QoR may improve if that partition is recompiled every time the RTL is recompiled, allowing those instances to adapt to changes in their neighbors. This behavior may even be a sign that the RTL should be re-architected to absorb those glue logic instances into the top-level block or their neighboring instances.

Tools also exist in the toolbar to zoom (see Zooming the Floorplanner In and Out (see page 312)) to the instances of the selected partition, search for the instances of the selected partition (this generates the appropriate Tcl find (see page 553) command to populate the Search View (see page 132)), and add the instances of the selected partition to the selection set (done with the Tcl select (see page 600) command, with results displayed in the Selection View (see page 136)).

The partition table includes filtering functionality, which can be used to control visibility of the partition rows. To enable filtering, click the ( ) **Toggle Filter Row Visibility** button. The filter row allows a content-appropriate filter to be applied to one or more table columns. Numeric columns filter based upon number ranges, while name columns filter based upon string matching or even regular expressions. For example, clicking the ( ) filter icon above the LUTs column allows viewing only partitions with, for example, greater than 3,000 LUTs. This filter functionality is most useful when there are a large number of partitions.

Tip

After using the ( ) Placement Region Tool in the Floorplanner view to create a new placement region, if using partitions, drag-and-drop a row from the Partitions view onto the newly created placement region (either the appropriate row of the table in the Placement Regions View (see page 120), or directly into the placement region painted in the Floorplanner view). This action generates the correct add_region_find_insts (see page 536) command to add all instances in the dragged partition to the designated placement region.

Netlist Browser

The Netlist Browser View (see page 89) also contains several features dedicated to the Incremental Compile Flow. There is a column (Partition) naming which partition owns all instances represented by that row in the table. No partition name is given if the instances are not owned by a partition, or if they are split between two or more partitions. When a partition is highlighted in the Partitions view, that highlight color is also present in the Highlight Color column of the Netlist Browser in all rows owned by the given partition.

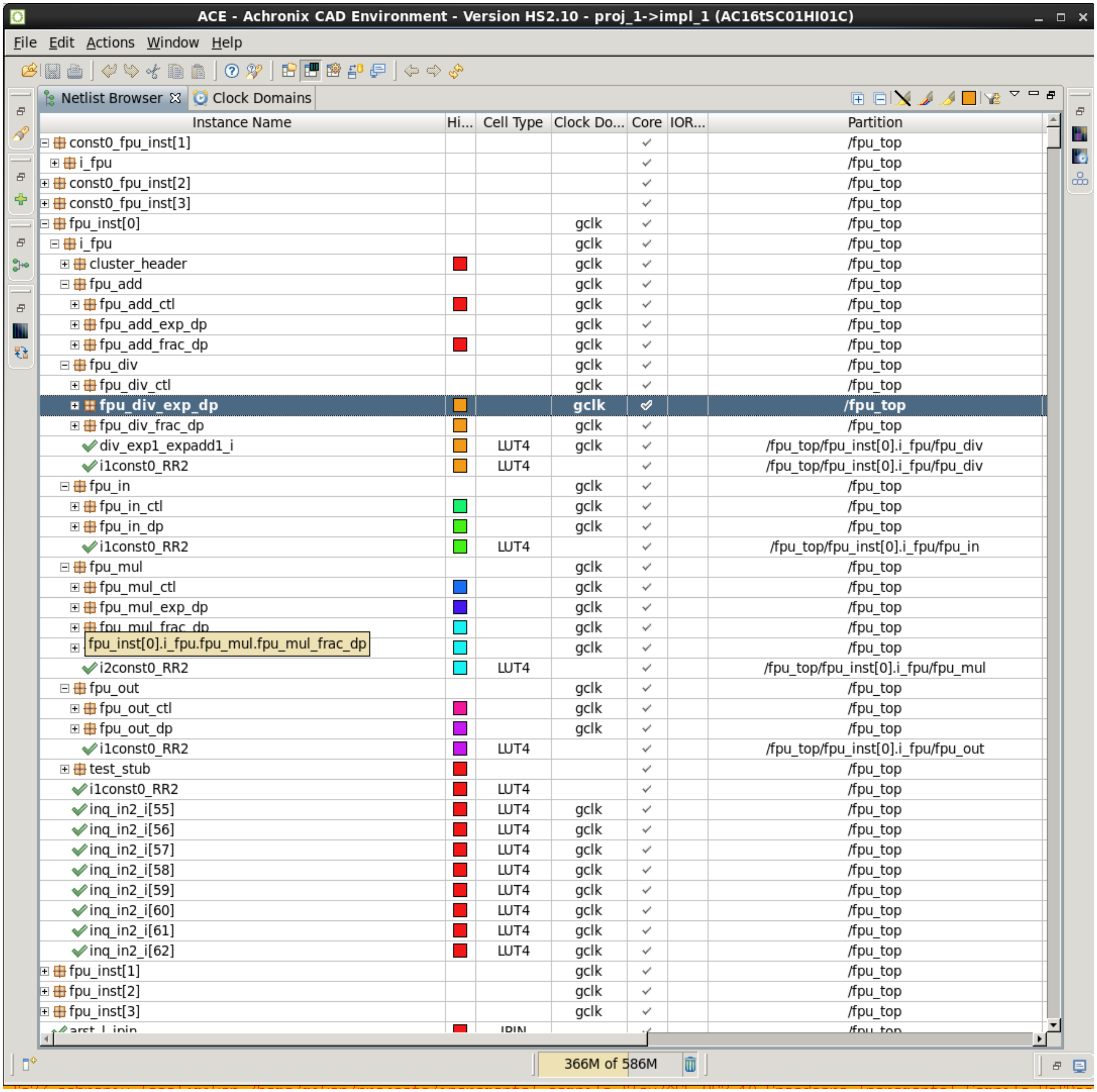


Figure 181: *ACE Netlist Browser*

End of the First Pass of ACE Place and Route

At this point, exit out of ACE if desired.

> **Note**
>
> ACE can stay in memory and "incremental" runs can be implemented in the same ACE session because ACE recognizes that inputs to the ACE flows have changed, and runs incremental flows appropriately.

Step 8: Change the RTL (rtl_V1)

It is at this point where the utility of the Incremental Compile Flow begins to become apparent. The next steps simulate the actions of a product development team in the middle of a design iteration by modifying the RTL for one of the partitions, and then rerunning Synthesis, Prepare, Placement, and Routing (SPP&R).

First, navigate to the top-level directory of the tutorial project and start Synplify-Pro if it is no longer running. Under Linux, execute:

```
% cd <your work area>/ Speedcore_Incremental_Compile_Reference_Design_RD012
% synplify_pro
```

Under Windows, double-click the Synplify Pro icon.

This step simulates an RTL change by replacing the source file `rtl/fpu_mul_ctl.v` with the version in the directory `rtl_V1`. To do this in Synplify Pro, in the Project Files tab, navigate to the Verilog folder and left click the file to be changed to select it by clicking its name, in this case `fpu_mul_ctl.v`, and then right-click to bring up popup menu followed by **Change File…** or select **Project** → **Change File**. In the dialog box that appears, use the drop-down box of the Look in field and navigate from the directory `rtl` to the directory `rtl_V1`. Then double-click the file `fpu_mul_ctl.v` , or select it and click **OK**. The old version of the file is then removed from the project and replaced by the modified version. Diff the old and new versions of the file to see the changes:

```
$ diff rtl_V1/fpu_mul_ctl.v rtl
```

> **Note**
>
> A flop has been added to the 5-bit wire `mul_exc_out`.

Step 9: Recompile the Design in Synplify Pro (rtl_V1)

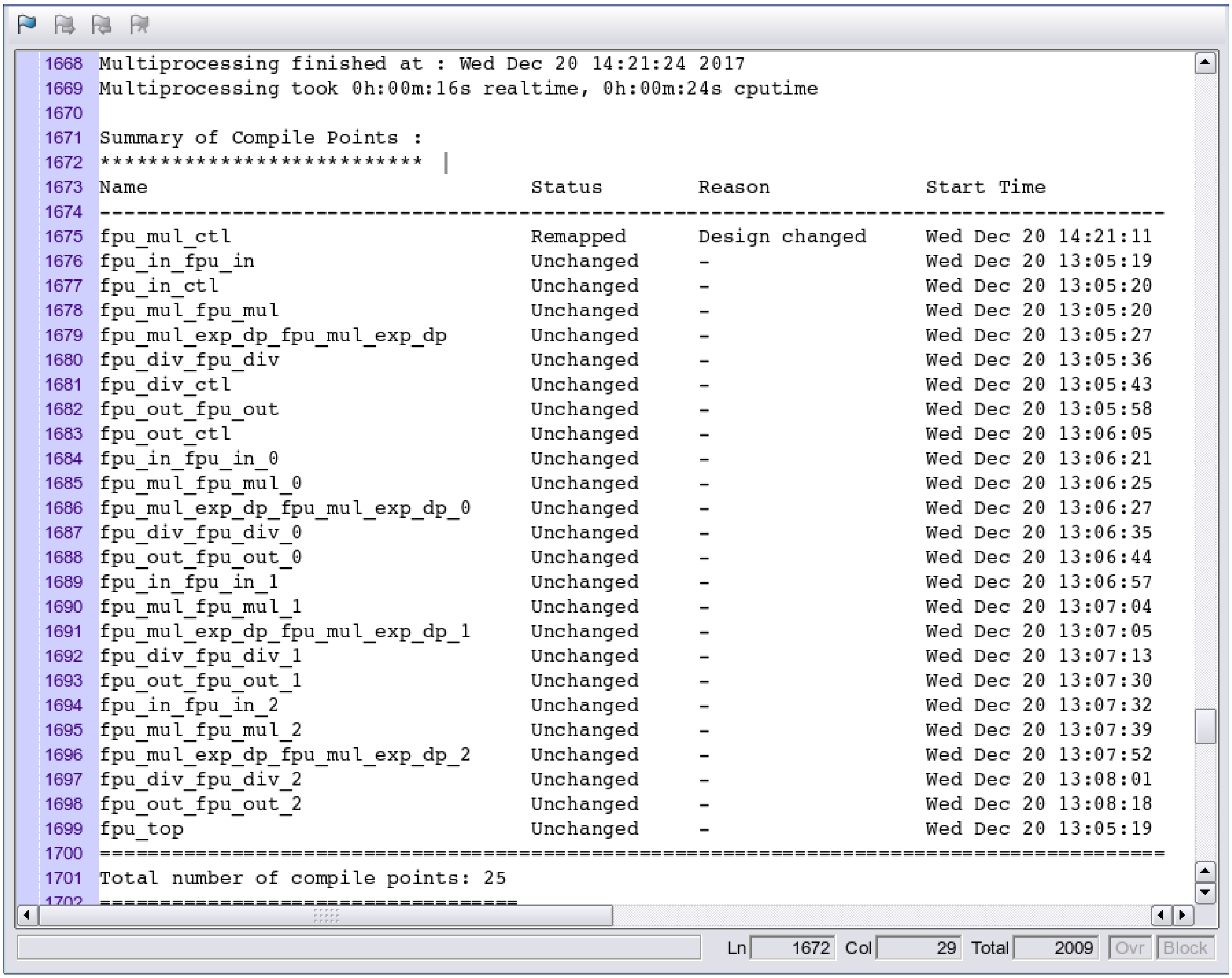
Click **Run** on the Synplify Pro home screen to recompile and remap the design.

> **Note**
>
> Runtime is much faster than in the first iteration because only the changed module needs to be recompiled.

Step 10: Review Synplify Results (rtl_V1)

Synplify Pro Log File (rtl_v1)

Once again, open the Synplify Pro log file `Speedcore_Incremental_Compile_Reference_Design_RD012/synplify/rev_1/open_sparc_fpu_icf_demo.srr` and search for the section titled "Summary of Compile Points". All of the partitions have a status of "unchanged" except for the partition `fpu_mul_ctl` and its parent partition, which have status "remapped" and a reason of "design changed". The timestamp of the remapped partitions have also been advanced.



```
Multiprocessing finished at : Wed Dec 20 14:21:24 2017
Multiprocessing took 0h:00m:16s realtime, 0h:00m:24s cputime

Summary of Compile Points :
***************************
Name                                 Status        Reason              Start Time
-------------------------------------------------------------------------------------------
fpu_mul_ctl                          Remapped      Design changed      Wed Dec 20 14:21:11
fpu_in_fpu_in                        Unchanged     -                   Wed Dec 20 13:05:19
fpu_in_ctl                           Unchanged     -                   Wed Dec 20 13:05:20
fpu_mul_fpu_mul                      Unchanged     -                   Wed Dec 20 13:05:20
fpu_mul_exp_dp_fpu_mul_exp_dp        Unchanged     -                   Wed Dec 20 13:05:27
fpu_div_fpu_div                      Unchanged     -                   Wed Dec 20 13:05:36
fpu_div_ctl                          Unchanged     -                   Wed Dec 20 13:05:43
fpu_out_fpu_out                      Unchanged     -                   Wed Dec 20 13:05:58
fpu_out_ctl                          Unchanged     -                   Wed Dec 20 13:06:05
fpu_in_fpu_in_0                      Unchanged     -                   Wed Dec 20 13:06:21
fpu_mul_fpu_mul_0                    Unchanged     -                   Wed Dec 20 13:06:25
fpu_mul_exp_dp_fpu_mul_exp_dp_0      Unchanged     -                   Wed Dec 20 13:06:27
fpu_div_fpu_div_0                    Unchanged     -                   Wed Dec 20 13:06:35
fpu_out_fpu_out_0                    Unchanged     -                   Wed Dec 20 13:06:44
fpu_in_fpu_in_1                      Unchanged     -                   Wed Dec 20 13:06:57
fpu_mul_fpu_mul_1                    Unchanged     -                   Wed Dec 20 13:07:04
fpu_mul_exp_dp_fpu_mul_exp_dp_1      Unchanged     -                   Wed Dec 20 13:07:05
fpu_div_fpu_div_1                    Unchanged     -                   Wed Dec 20 13:07:13
fpu_out_fpu_out_1                    Unchanged     -                   Wed Dec 20 13:07:30
fpu_in_fpu_in_2                      Unchanged     -                   Wed Dec 20 13:07:32
fpu_mul_fpu_mul_2                    Unchanged     -                   Wed Dec 20 13:07:39
fpu_mul_exp_dp_fpu_mul_exp_dp_2      Unchanged     -                   Wed Dec 20 13:07:52
fpu_div_fpu_div_2                    Unchanged     -                   Wed Dec 20 13:08:01
fpu_out_fpu_out_2                    Unchanged     -                   Wed Dec 20 13:08:18
fpu_top                              Unchanged     -                   Wed Dec 20 13:05:19
===========================================================================================
Total number of compile points: 25
====================================
```


Figure 182: ***Synplify Pro Log File Showing Changed Compile Points***

ACE Partitioning Constraints File (rtl_V1)

Also re-open file `Speedcore_Incremental_Compile_RefDesign_RD012/synplify/rev_1/open_sparc_fpu_icf_demo_partition.prt` and observe that the same changes are also reflected in the constraints file written out for ACE. Again, the timestamp has advanced when compared with the unmodified partitions.

```
set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1513803914" -cp_type "hard"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
```

Ln 25 Col 63 Total 37 Ovr Block

Figure 183: *ACE Partitioning Constraints File: open_sparc_fpu_icf_demo_partition.prt*

Use **File** → **Close** to close Synplify Pro, and click **Save changes to project proj_1**.

Step 11: Recompile the Design in ACE (rtl_V1)

If ACE was exited earlier, navigate to the same ace directory as before, and start ACE again. Otherwise, continue on from the current ACE session.

Ensure that this tutorial project is the active project. Again, uncheck the **Run Sign-off Timing Analysis** and **Generate Bitstream** flow steps to save some runtime.

Click the green triangle ( ) in the upper-right corner of the Flow view to rerun the Prepare, Placement, and Routing flow.

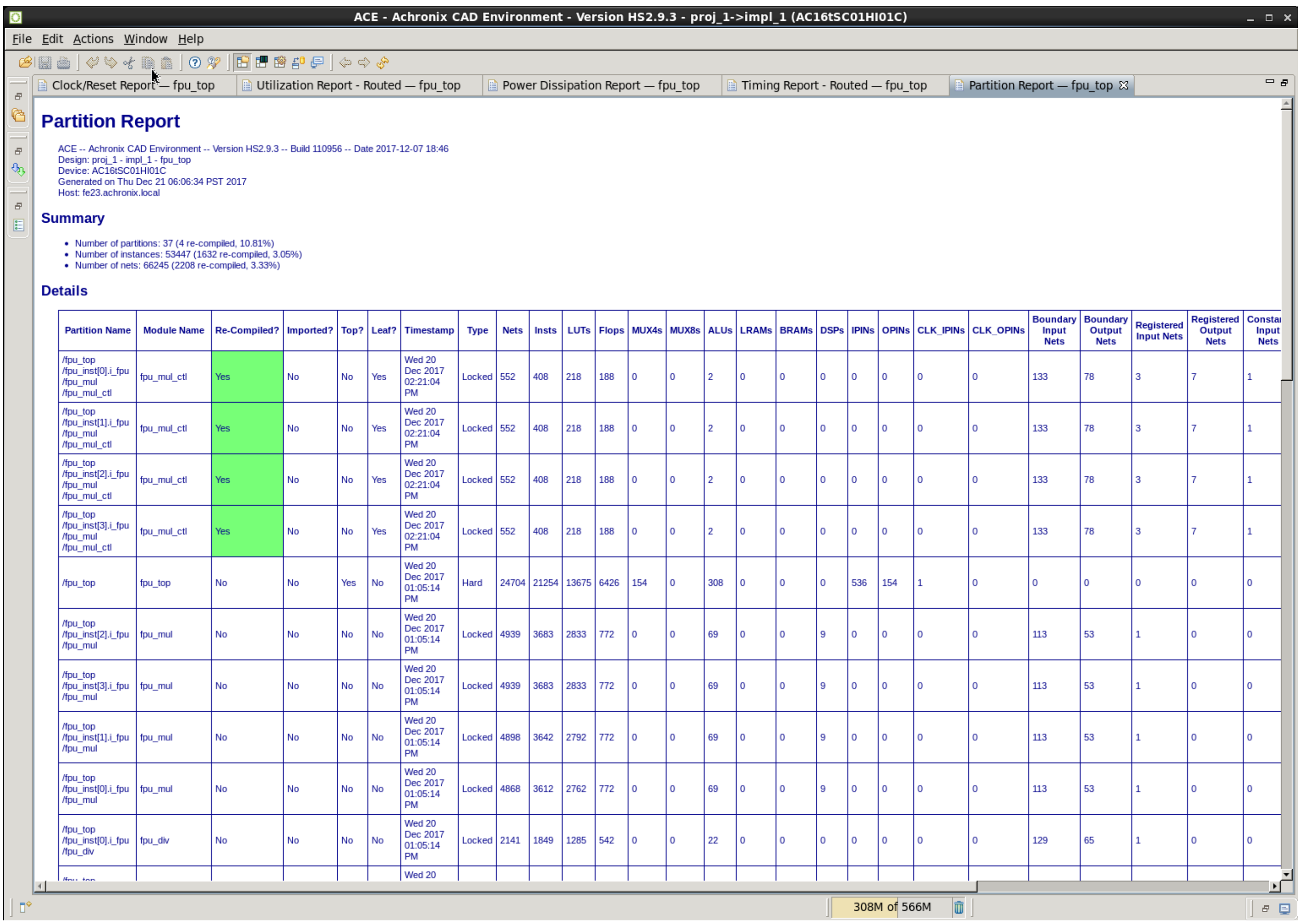
> **Note**
>
> The runtime of the flow is also significantly reduced over the first non-incremental compile.

In this second pass, ACE reads the new `open_sparc_fpu_icf_demo.vm` netlist file and the new `open_sparc_fpu_icf_demo_partition.prt` constraints file from the synplify directory. During the run_prepare flow step, ACE then executes an operation called Tear & Stitch. Each partition which has not been recompiled during synthesis is *torn* out of the database, and a copy from the previous pass is *stitched* back in. The copy from the previous run contains the complete set of placement and routing data. The placement of all stitched instances are locked, and all routes are marked as preroutes to prevent their modification when the remainder of the netlist is placed and routed.

Step 12: Review ACE Results (rtl_V1)

Partition Report (rtl_V1)

Maximize the Partition Report tab in the Editor Area of the Projects perspective. In the summary section, only 4 of the 37 partitions (10.81%) were recompiled by ACE, resulting in 3.05% of the instances being re-placed and 3.33% of the nets being rerouted. Also note in the Details section that only 4 fpu_mul_ctl partitions were recompiled, and their new timestamps are displayed. The counts of instances and nets in those partitions have changed by a small amount.



Partition Report

ACE -- Achronix CAD Environment -- Version HS2.9.3 -- Build 110956 -- Date 2017-12-07 18:46
Design: proj_1 - impl_1 - fpu_top
Device: AC16tSC01HI01C
Generated on Thu Dec 21 06:06:34 PST 2017
Host: fe23.achronix.local

Summary

- Number of partitions: 37 (4 re-compiled, 10.81%)
- Number of instances: 53447 (1632 re-compiled, 3.05%)
- Number of nets: 66245 (2208 re-compiled, 3.33%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Consta Input Nets
/fpu_top /fpu_inst[0].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[1].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[2].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[3].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4898	3642	2792	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4868	3612	2762	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	fpu_div	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0

Figure 184: *ACE Partition Report After the rtl_V1 Incremental Compile Iteration*

Floorplanner View (rtl_V1)

Switch to the ( ) Floorplanner perspective. In the Floorplanner view, compared to figure above (see page 386) from the first iteration, about 3% fewer instances are now drawn with a locked fill color (dark yellow by default).

The locked placement state is just one of several potential placement states (see Instance States (see page 243)) that an instance can have when painted in the Floorplanner. This locked placement state is somewhat similar in concept to the fixed placement state. (Instances with fixed placement status, shown in the Floorplanner with a light yellow fill color by default, are instances with user-assigned placement constraints, usually defined in a `.pdc` file. These fixed instances are not allowed to be moved from their constrained placements during the flow.) The locked placement state indicates instances that are locked in place because they are in a partition that was not recompiled. Only instances with the default (or soft) placement state (light-grey fill color by default) were allowed to have their placement changed during the flow.

> The visibility of the instance locked placement state and the associated locked fill color can be chosen within the ACE GUI preferences. To see/edit these, select **Window** → **Preferences** → **Floorplanner View Colors**. In particular, ensure the **Instances** → **Show Locked Color on Instances with Locked Placement** box is checked, and that the **Locked Instances View Color** is set to the desired color (dark yellow by default).

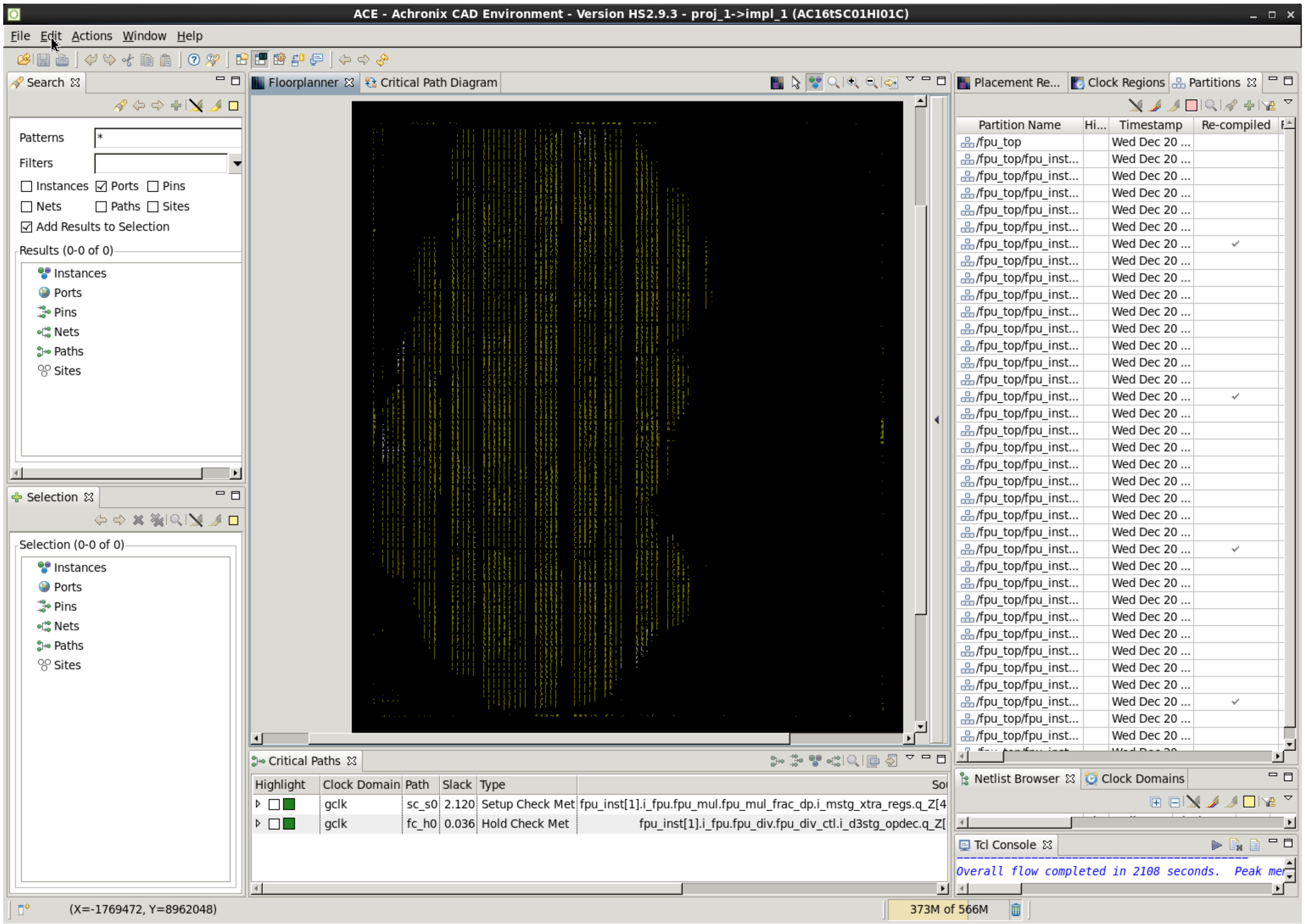


Figure 185: ***ACE Floorplanner Perspective After the rtl_V1 Incremental Compile Iteration***

To get a feel for the amount of re-routing that was required, select (add to the ACE Selection Set) the instances in the partitions that were recompiled and then view the flylines representing the nets for those instances. To do this, check the **Selected Instance Flylines** box in the Floorplanner view fly-out palette. Then in the Partitions view, choose the rows for the partitions that were recompiled, and click ( ) **Add Instances to Selection** in the Partitions View toolbar. Blue flylines are drawn for the nets of the selected instances (or rather for the first 200 selected instances, since the selection set contains more than 200 objects). In the Selection View (see page 136) click the gold left-arrow ( ) and right-arrow ( ) buttons in the view header to cycle visibility between different subsets of 200 of the selection set instances at a time.

> **Note**
>
> There are many more than 200 nets that are actually re-routed. However, by using the selection set to filter the visible connectivity in this manner, the amount of visible change to the design has been minimized to just the instances of the module that was changed.

This information is helpful in understanding the effect of any placement and routing changes during the second pass. Instances in a failing critical path with wayward placement could indicate that changes in the RTL were too extensive for effective incremental recompilation. This situation can occur when one of the partitions grows significantly in size and no longer fits in the area between locked neighboring partitions, for example. The placement for this recompiled partition may be squeezed into an undesirable aspect ratio, forcing long routing detours. In situations such as this, it may be best to force the entire design to be recompiled by enabling **Force Recompile on Next Run** for all of the partitions in the Partitions View.

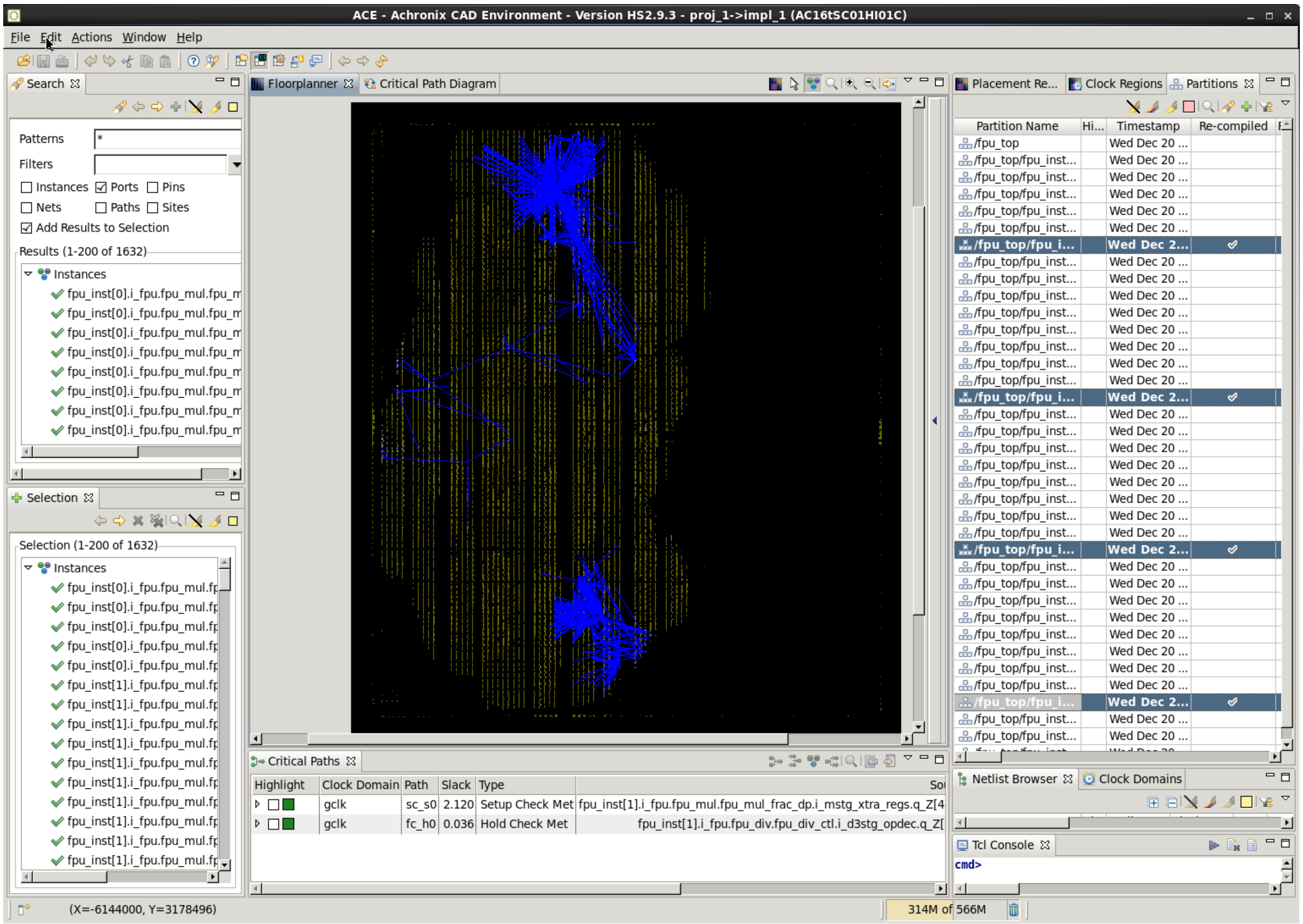


Figure 186: *ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V1)*

Partitions View (rtl_V1)

In the Partitions View (see page 117) of the ( ) Floorplanner Perspective, note that only 4 of the 37 partitions have a check mark in the Re-Compiled column.

Click ( ) **Deselect all** in the Selection view to remove the routing flylines, and then click ( ) **Auto-Highlight Partitions** in the Partitions View to observe any changes in the placement of the changed partitions. Unchanged partitions can also be manually unhighlighted to only see (highlight) those that were not re-placed. This technique can help in understanding the effectiveness of incremental compilation on the design.

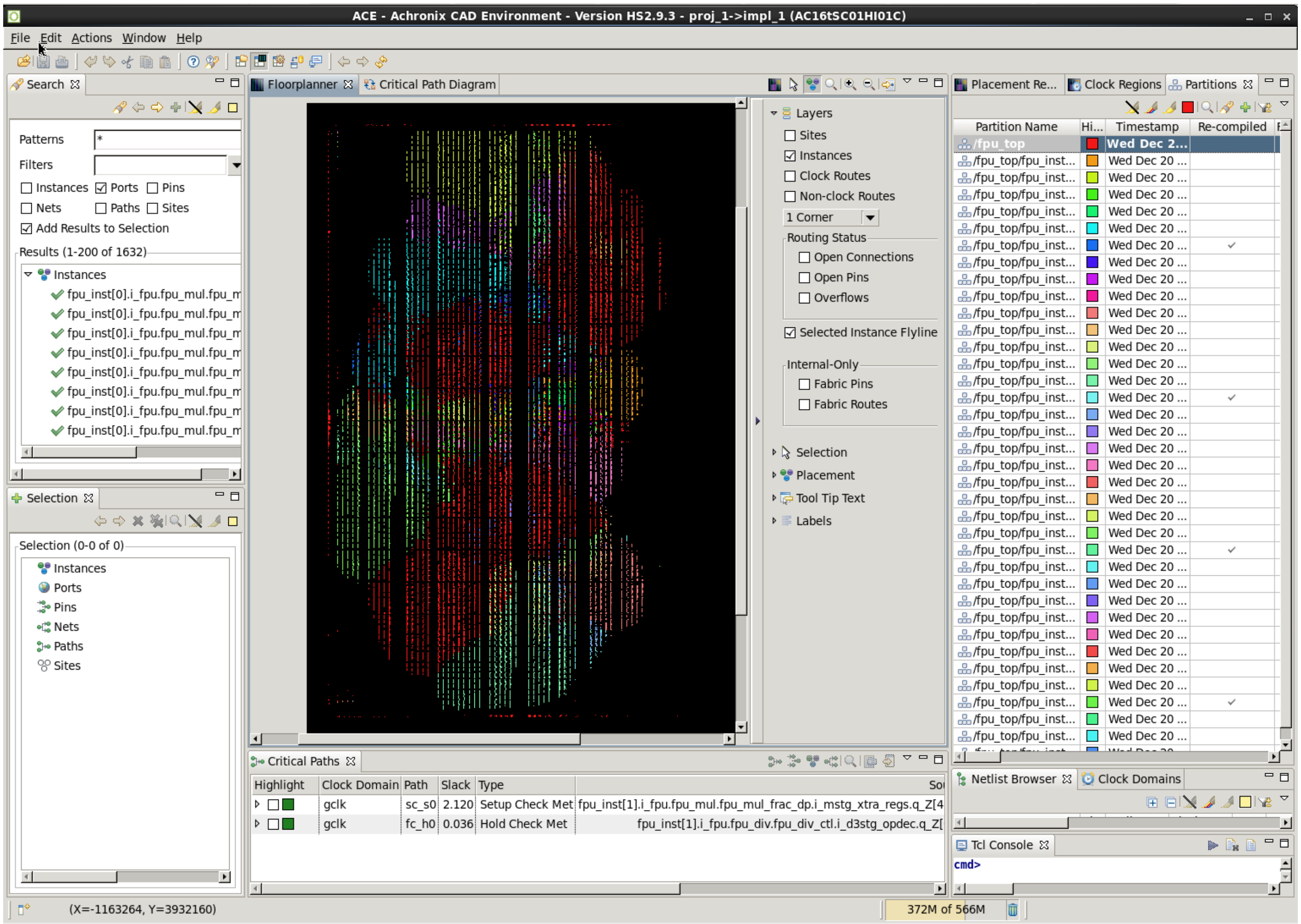


Figure 187: ***Highlighting Partitions in ACE After Second Incremental Compile Iteration***

Step 13: Additional Incremental Iterations

Steps 8–12 can be rerun with additional RTL changes if desired. This tutorial design contains additional directories with additional RTL change examples. Or modify the existing RTL files by adding or deleting module pins, add or remove partitions from the .fdc file, or rename module instances, to further explore the incremental compile flow.

Table 151: *Additional Tutorial Examples*

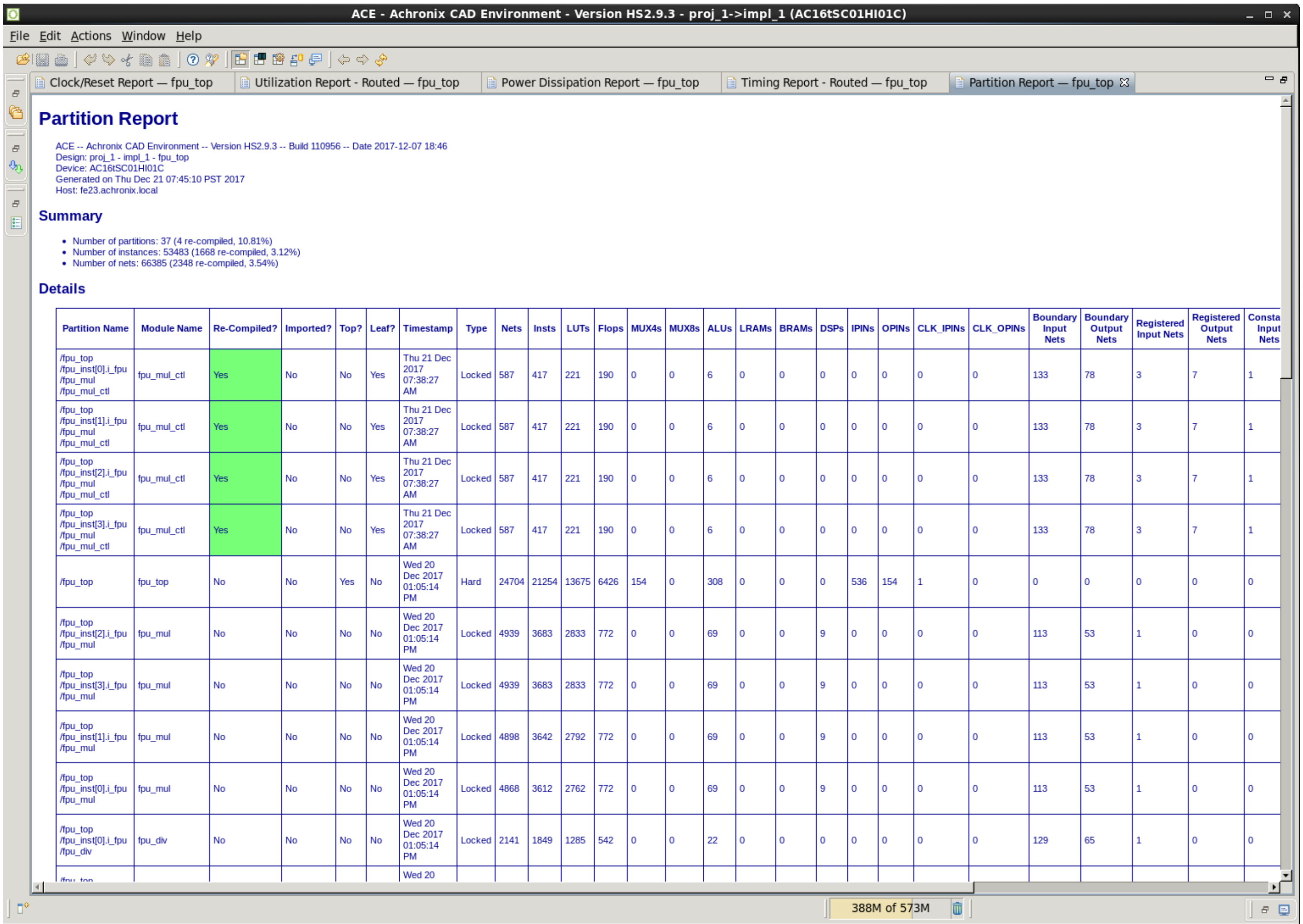
Directory	Changes
`rtl_V1`	Flopped the signal mul_exc_out in fpu_mul_ctl.v
`rtl_V2`	Reverts the changes from rtl_V1, adds a new 6-bit counter in place of a constant
`rtl_V3`	Modifies the partition fpu_div_ctl by adding an enable check
`rtl_V4`	Modifies the top-level partition fpu.v by inverting one net

Step 14: Review ACE Results (rtl_V2)

After rerunning the synplify_pro and ace flows, review the perturbation of the design in ace by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V2)

Maximize the Partition Report tab in the Editor Area of the Projects perspective. In the summary section, only 4 of the 37 partitions (10.81%) were recompiled by ACE, resulting in 3.12% of the instances being re-placed and 3.54% of the nets being rerouted. Also note in the Details section that only 4 fpu_mul_ctl partitions were recompiled, and their new timestamps are displayed. The counts of instances and nets in those partitions have changed by a small amount.



Partition Report

ACE -- Achronix CAD Environment -- Version HS2.9.3 -- Build 110956 -- Date 2017-12-07 18:46
Design: proj_1 - impl_1 - fpu_top
Device: AC16tSC01HI01C
Generated on Thu Dec 21 07:45:10 PST 2017
Host: fe23.achronix.local

Summary

- Number of partitions: 37 (4 re-compiled, 10.81%)
- Number of instances: 53483 (1668 re-compiled, 3.12%)
- Number of nets: 66385 (2348 re-compiled, 3.54%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Consta Input Nets
/fpu_top /fpu_inst[0].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 07:38:27 AM	Locked	587	417	221	190	0	0	6	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[1].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 07:38:27 AM	Locked	587	417	221	190	0	0	6	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[2].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 07:38:27 AM	Locked	587	417	221	190	0	0	6	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top /fpu_inst[3].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 07:38:27 AM	Locked	587	417	221	190	0	0	6	0	0	0	0	0	0	0	133	78	3	7	1
/fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4898	3642	2792	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4868	3612	2762	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	fpu_div	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0

Figure 188: *ACE Partition Report After the rtl_V2 Incremental Compile Iteration*

Floorplanner View (rtl_V2)

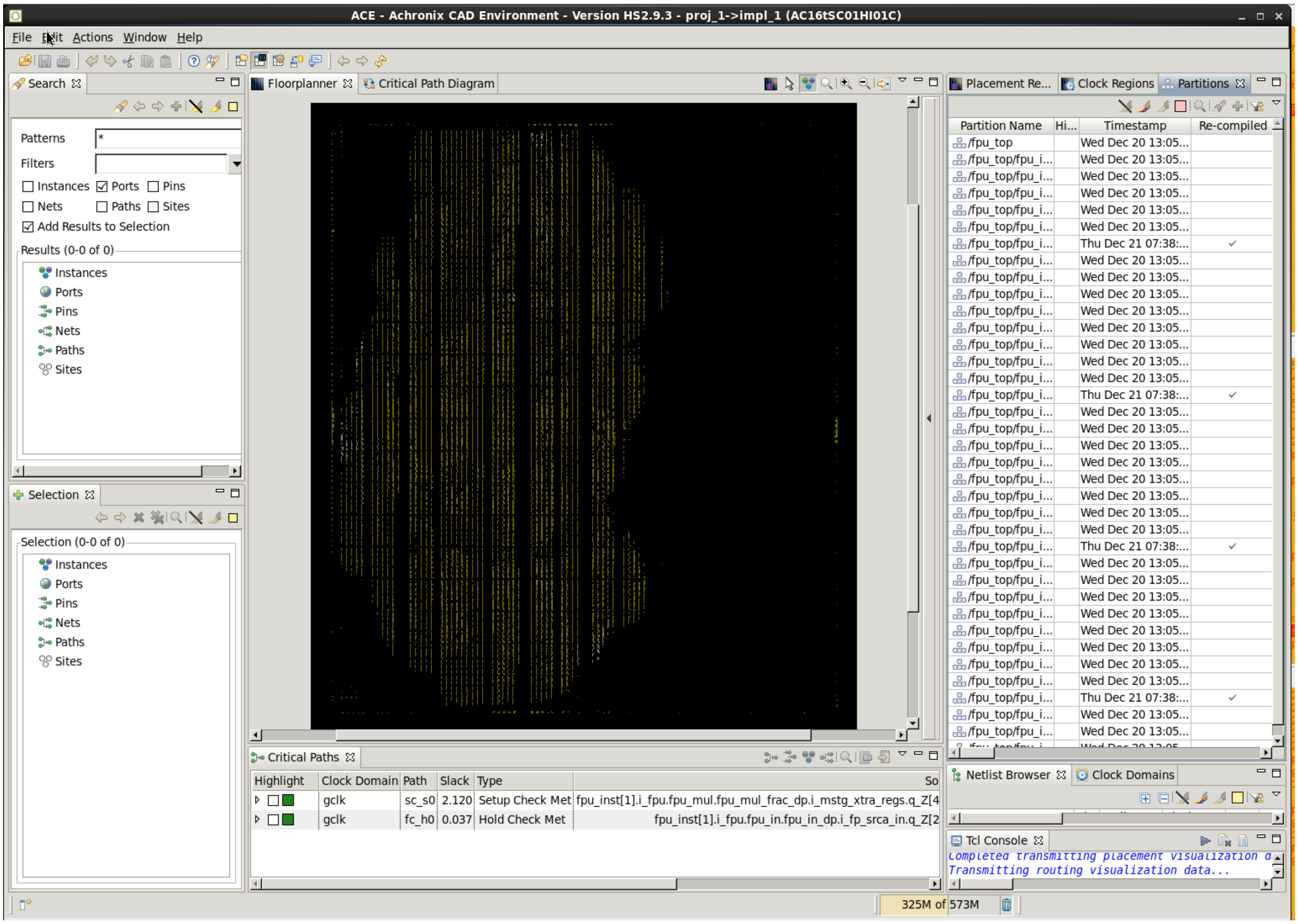


Figure 189: ***ACE Floorplanner Perspective After rtl_V2 Incremental Compile Iteration***

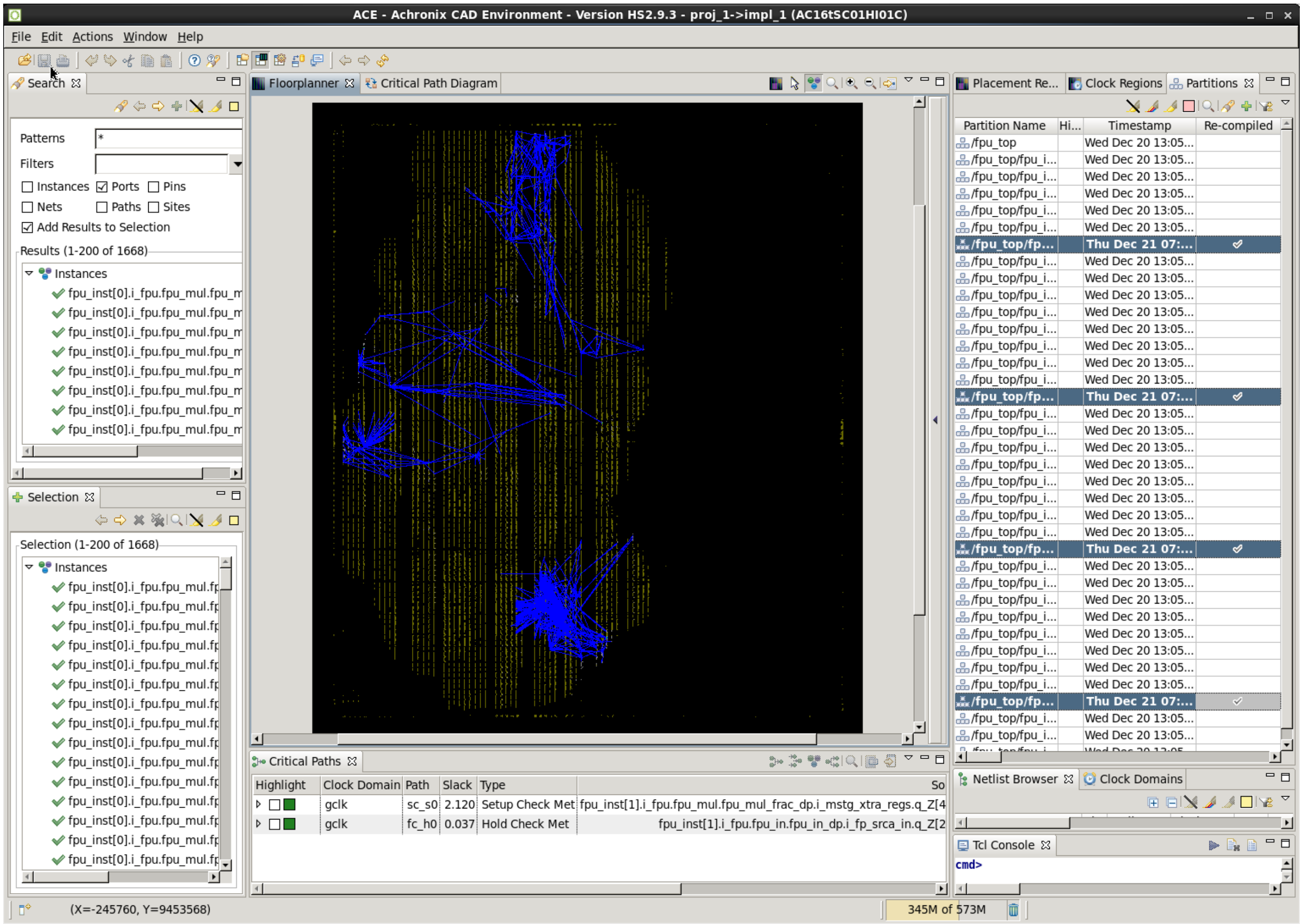
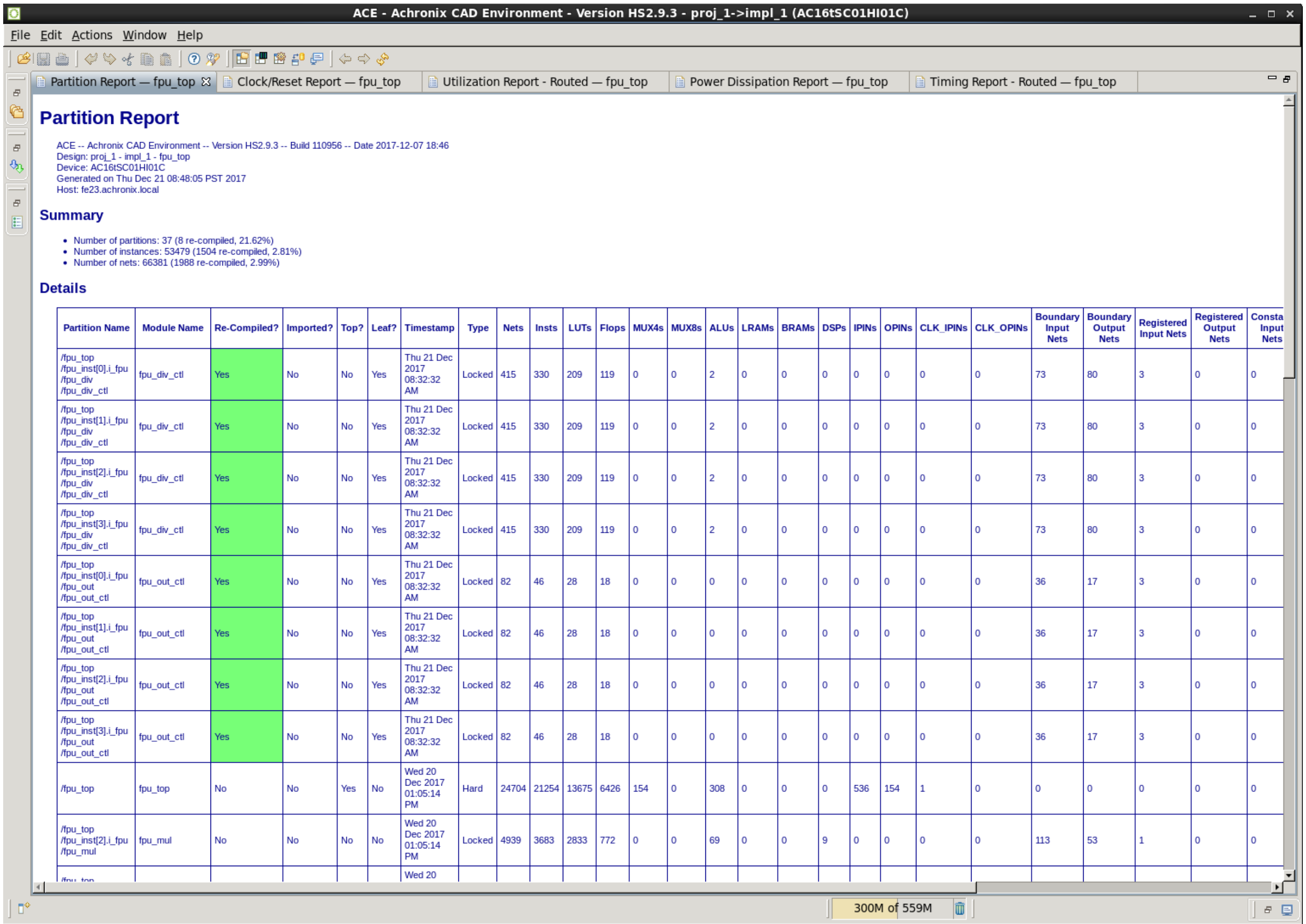


Figure 190: ***ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V2)***

Step 15: Review ACE Results (rtl_V3)

After rerunning the synplify_pro and ace flows, review the perturbation of the design in ace by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V3)



Partition Report

ACE -- Achronix CAD Environment -- Version HS2.9.3 -- Build 110956 -- Date 2017-12-07 18:46
Design: proj_1 - impl_1 - fpu_top
Device: AC16tSC01HI01C
Generated on Thu Dec 21 08:48:05 PST 2017
Host: fe23.achronix.local

Summary

- Number of partitions: 37 (8 re-compiled, 21.62%)
- Number of instances: 53479 (1504 re-compiled, 2.81%)
- Number of nets: 66381 (1988 re-compiled, 2.99%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Consta Input Nets
/fpu_top /fpu_inst[0].i_fpu /fpu_div /fpu_div_ctl	fpu_div_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	73	80	3	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_div /fpu_div_ctl	fpu_div_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	73	80	3	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_div /fpu_div_ctl	fpu_div_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	73	80	3	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_div /fpu_div_ctl	fpu_div_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	73	80	3	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top						Wed 20																				

Figure 191: ***ACE Partition Report After the rtl_V3 Incremental Compile Iteration***

Floorplanner View (rtl_V3)

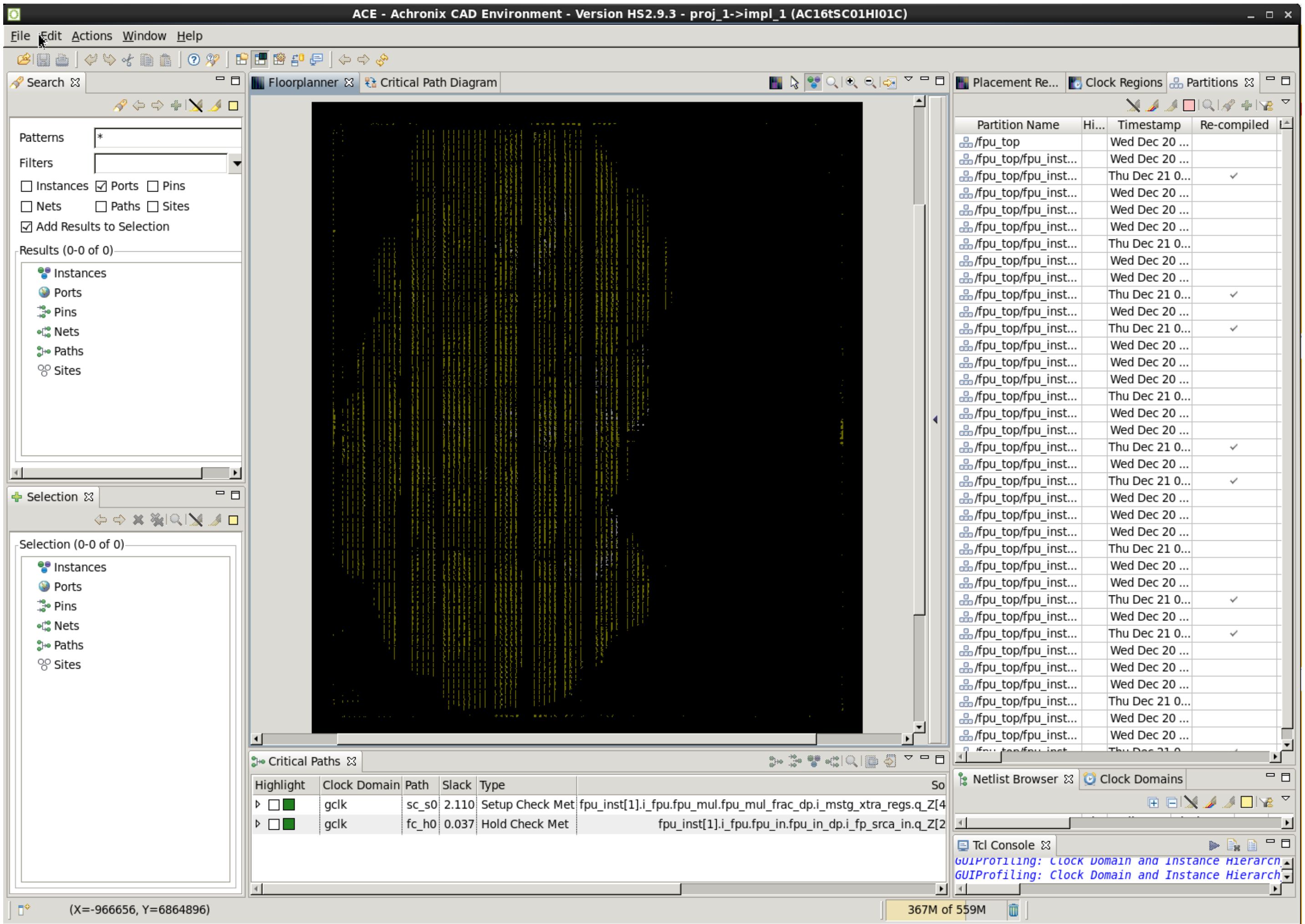


Figure 192: ***ACE Floorplanner Perspective After rtl_V3 Incremental Compile Iteration***

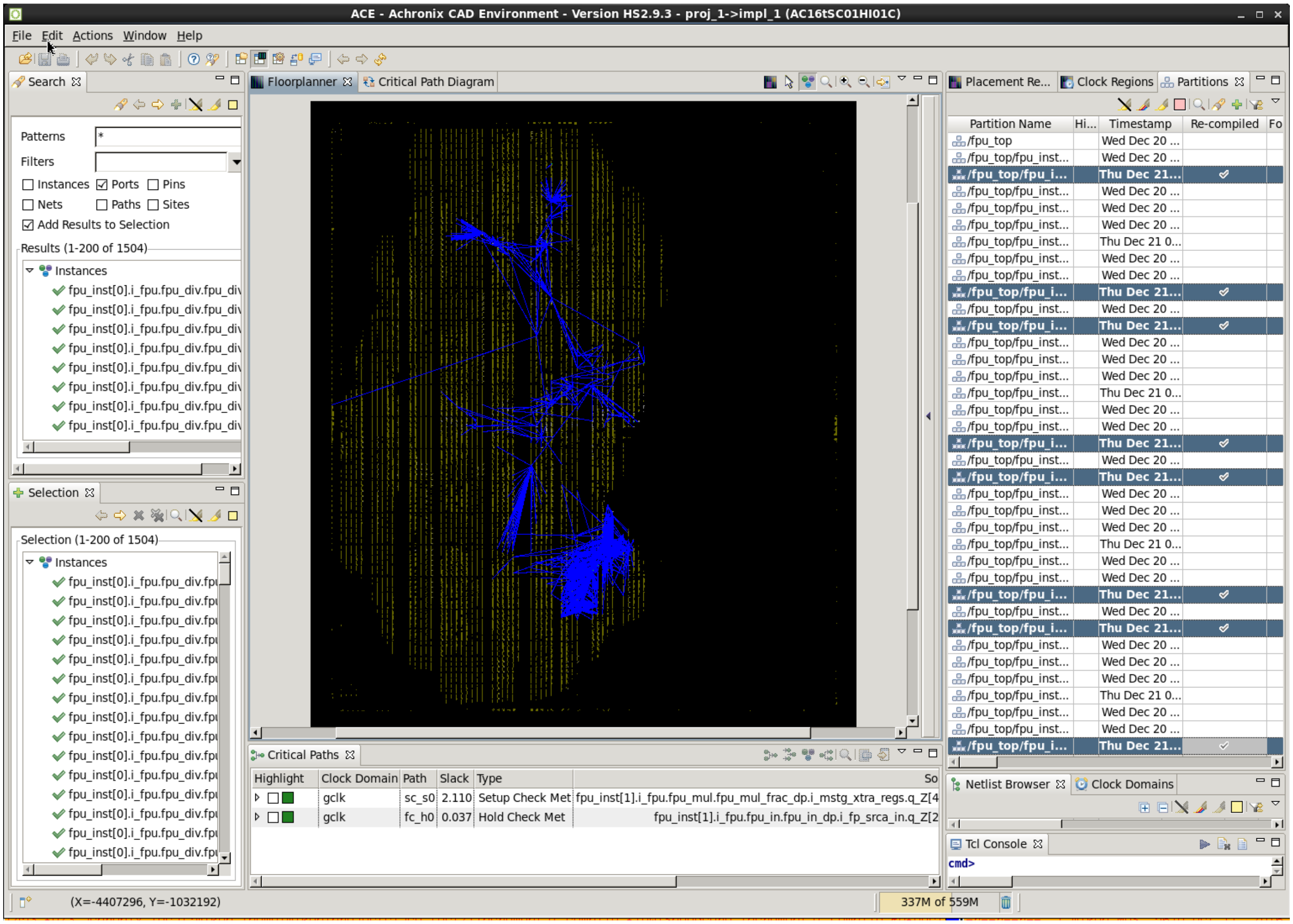
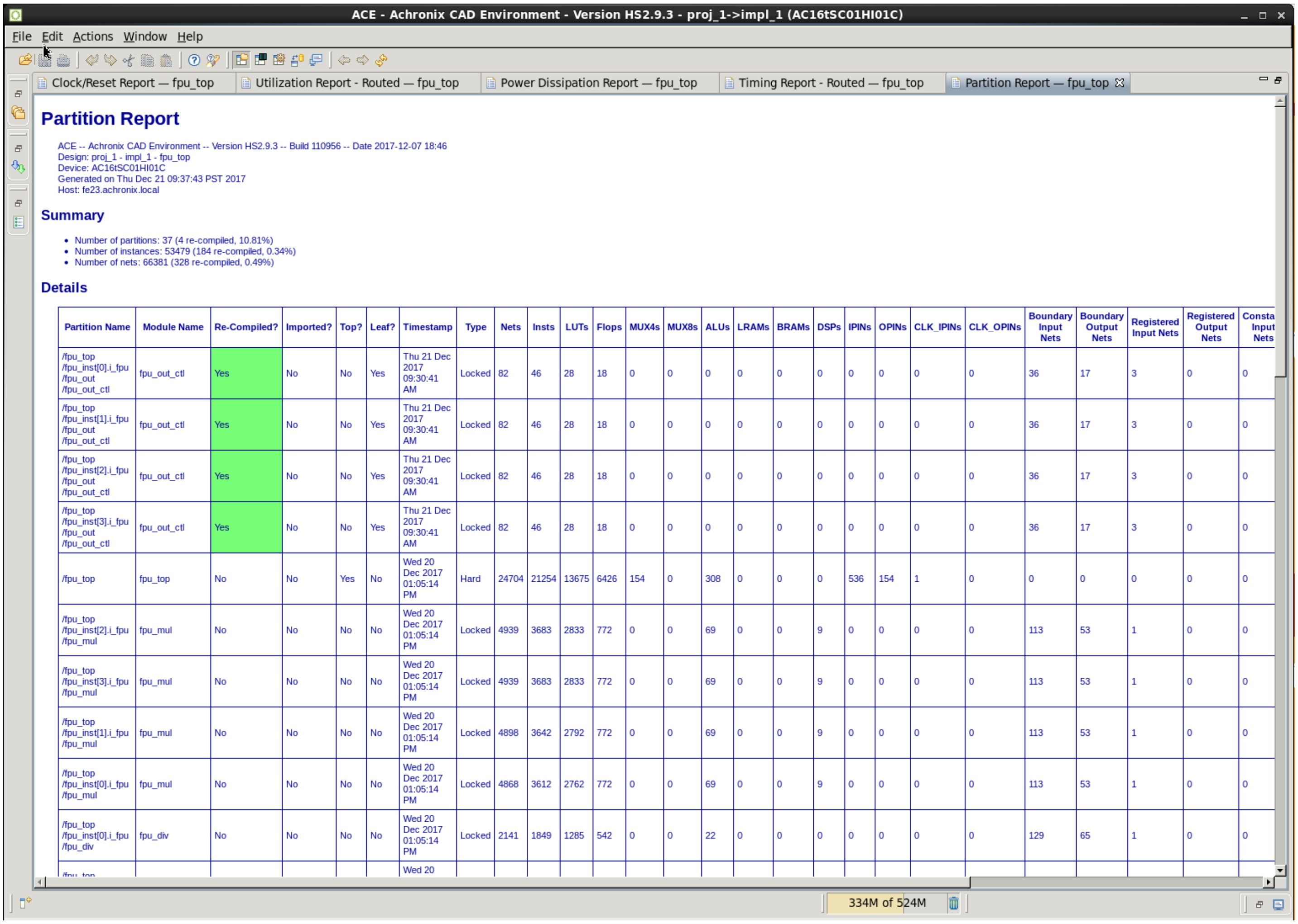


Figure 193: ***ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V3)***

Step 16: Review ACE Results (rtl_V4)

After rerunning the synplify_pro and ACE flows, review the perturbation of the design in ACE by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V4)



ACE - Achronix CAD Environment - Version HS2.9.3 - proj_1->impl_1 (AC16tSC01HI01C)

File Edit Actions Window Help

Clock/Reset Report — fpu_top | Utilization Report - Routed — fpu_top | Power Dissipation Report — fpu_top | Timing Report - Routed — fpu_top | Partition Report — fpu_top

Partition Report

ACE -- Achronix CAD Environment -- Version HS2.9.3 -- Build 110956 -- Date 2017-12-07 18:46
Design: proj_1 - impl_1 - fpu_top
Device: AC16tSC01HI01C
Generated on Thu Dec 21 09:37:43 PST 2017
Host: fe23.achronix.local

Summary

- Number of partitions: 37 (4 re-compiled, 10.81%)
- Number of instances: 53479 (184 re-compiled, 0.34%)
- Number of nets: 66381 (328 re-compiled, 0.49%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s
/fpu_top /fpu_inst[0].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_out /fpu_out_ctl	fpu_out_ctl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0
/fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4898	3642	2792	772	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4868	3612	2762	772	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	fpu_div	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0
/fpu_top						Wed 20							

Partition Name	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Consta Input Nets
/fpu_top /fpu_inst[0].i_fpu /fpu_out /fpu_out_ctl	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_out /fpu_out_ctl	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_out /fpu_out_ctl	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_out /fpu_out_ctl	0	0	0	0	0	0	0	0	36	17	3	0	0
/fpu_top	308	0	0	0	536	154	1	0	0	0	0	0	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	69	0	0	9	0	0	0	0	113	53	1	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	22	0	0	0	0	0	0	0	129	65	1	0	0

334M of 524M

Figure 194: ***ACE Partition Report After rtl_V4 Incremental Compile Iteration***

Floorplanner View (rtl_V4)

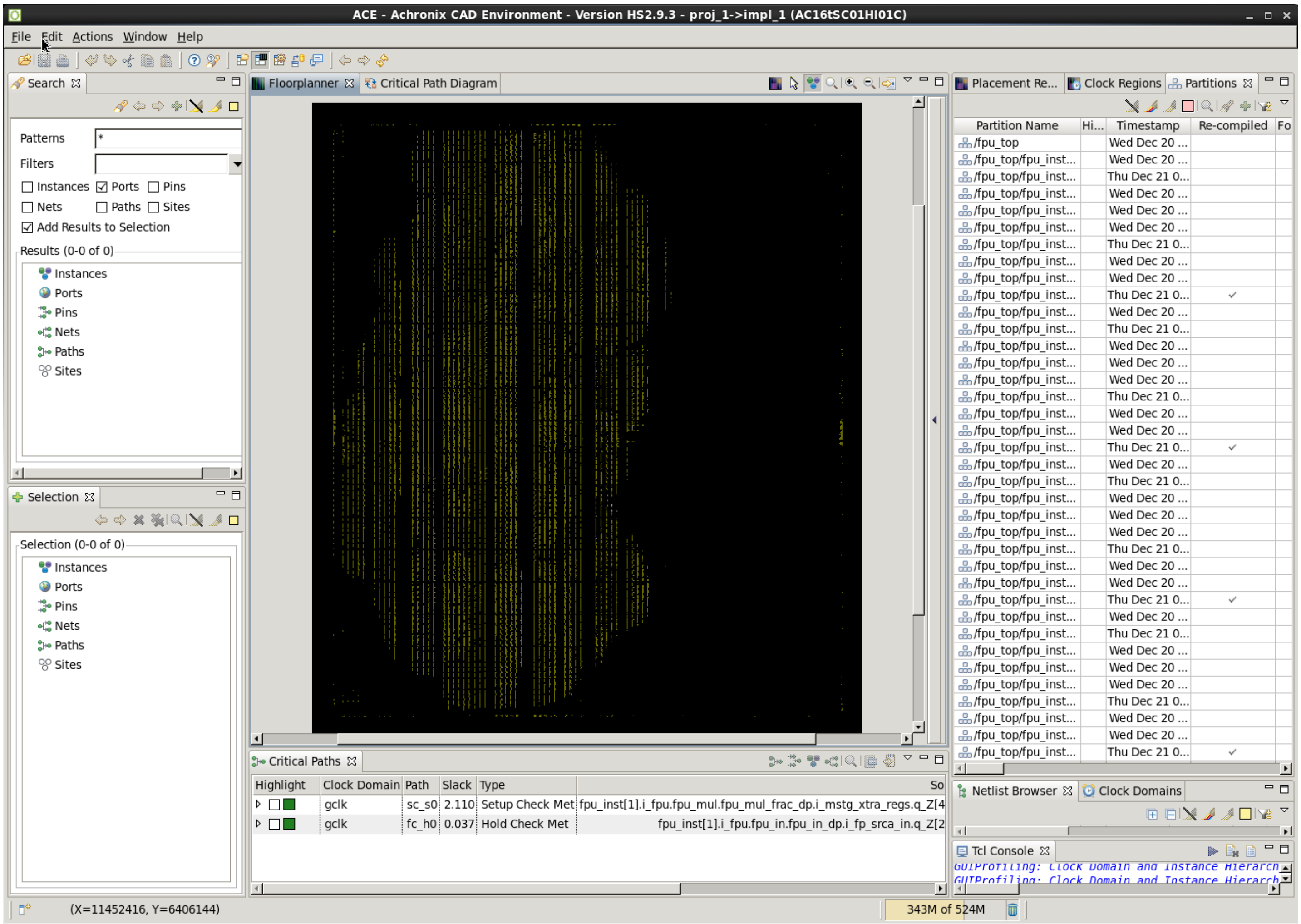


Figure 195: ***ACE Floorplanner Perspective After rtl_V4 Incremental Compile Iteration***

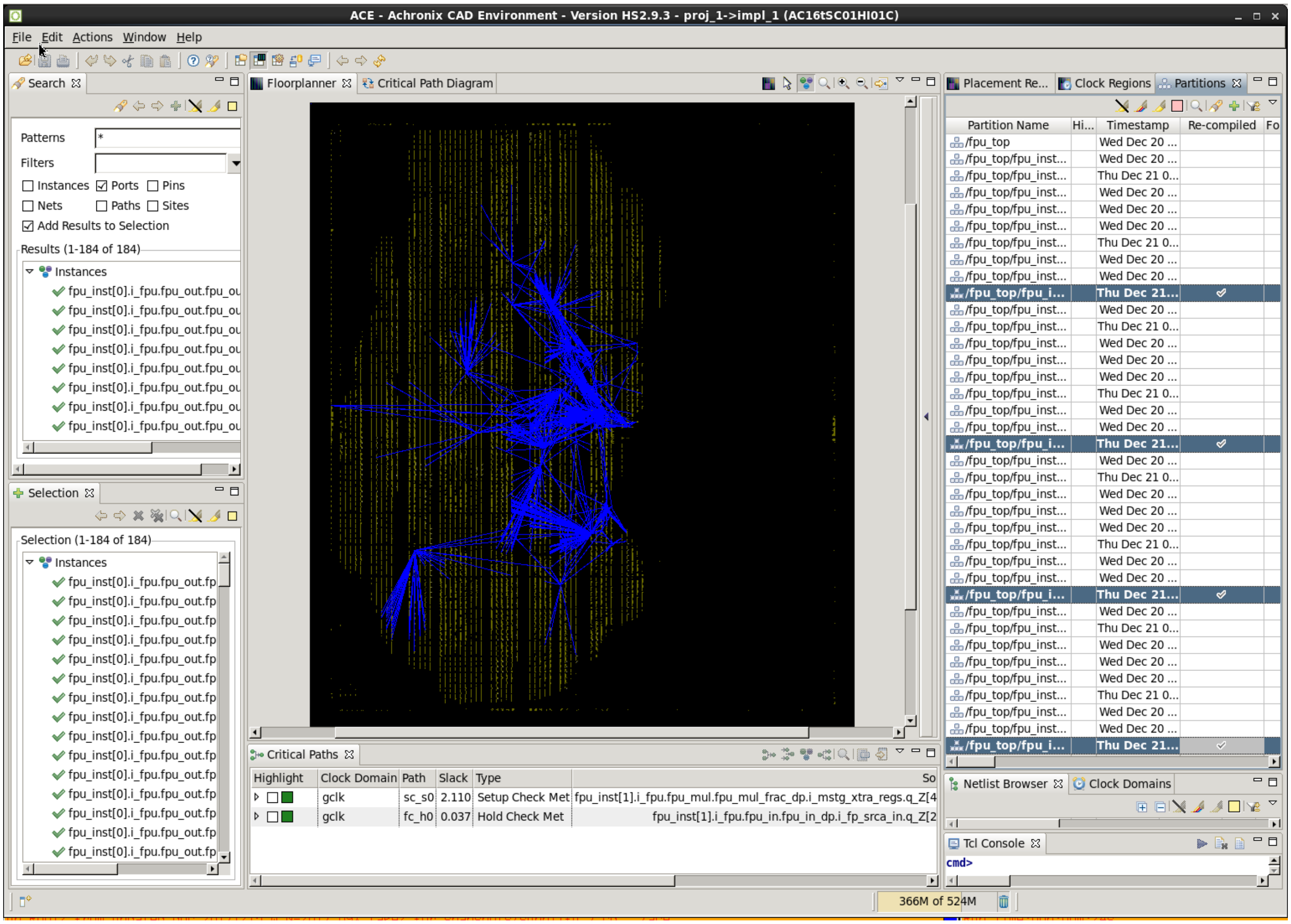


Figure 196: ***ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V4)***

> **Note**
>
> For details how to run a set of changes in order to select an optimal implementation, continue on the Multiprocess Incremental Compile Tutorial (see page 407). This second tutorial expands upon concepts from this tutorial.

Multiprocess Incremental Compile Tutorial

Using the incremental compile flow with the multiprocess GUI can be a powerful combination to help with timing closure. The Multiprocess GUI is used to try multiple experiments with different implementation options and/or sets of design constraints. The best implementation can then be selected to be the source for unchanged partitions in a subsequent incremental run. Across all implementations, the locked placement and routing data for all unchanged partitions is then merged from that best implementation. This merging is accomplished by copying the `best_impl/output/<design>.icdb` file from the best implementation to all other implementations before starting the next incremental iteration. All file copying is handled automatically by the multiprocess GUI. If you have not yet completed the Single-Process Incremental Compile Tutorial (see page 367), please complete Steps 1 through 5 of that tutorial now before proceeding.

Below are the step-by-step actions of using the multiprocess GUI inside the incremental compile flow.

Step 1: Compile the Design in Synplify Pro or Clear the ACE Project

If you have previously completed the Single-Process Incremental Compile Tutorial (see page 367), clear the ACE project and begin again from the beginning. Clear the project by deleting all of the files and subdirectories under the Ace directory of the tutorial work area. Otherwise, the first time ACE is run, it performs an incremental compile and the results do not match those described below.

Step 2: Create Multiprocess Implementations and Run ACE

From the ACE Home Screen, Use **Window** → **Show View** → **Multiprocess** to open the multiprocess GUI. Then select the **Generate Implementations from Option Sets** radio button. This action generates a large set of implementations automatically using a number of predefined implementation option variables. Optionally, set the “Parallel Job Count” option and configure the job submission system in the Ace preferences. Next click the **RUN** button (three stacked green triangles) to start running all implementations in parallel in the background. See the following screenshot of the Multiprocess View menus.

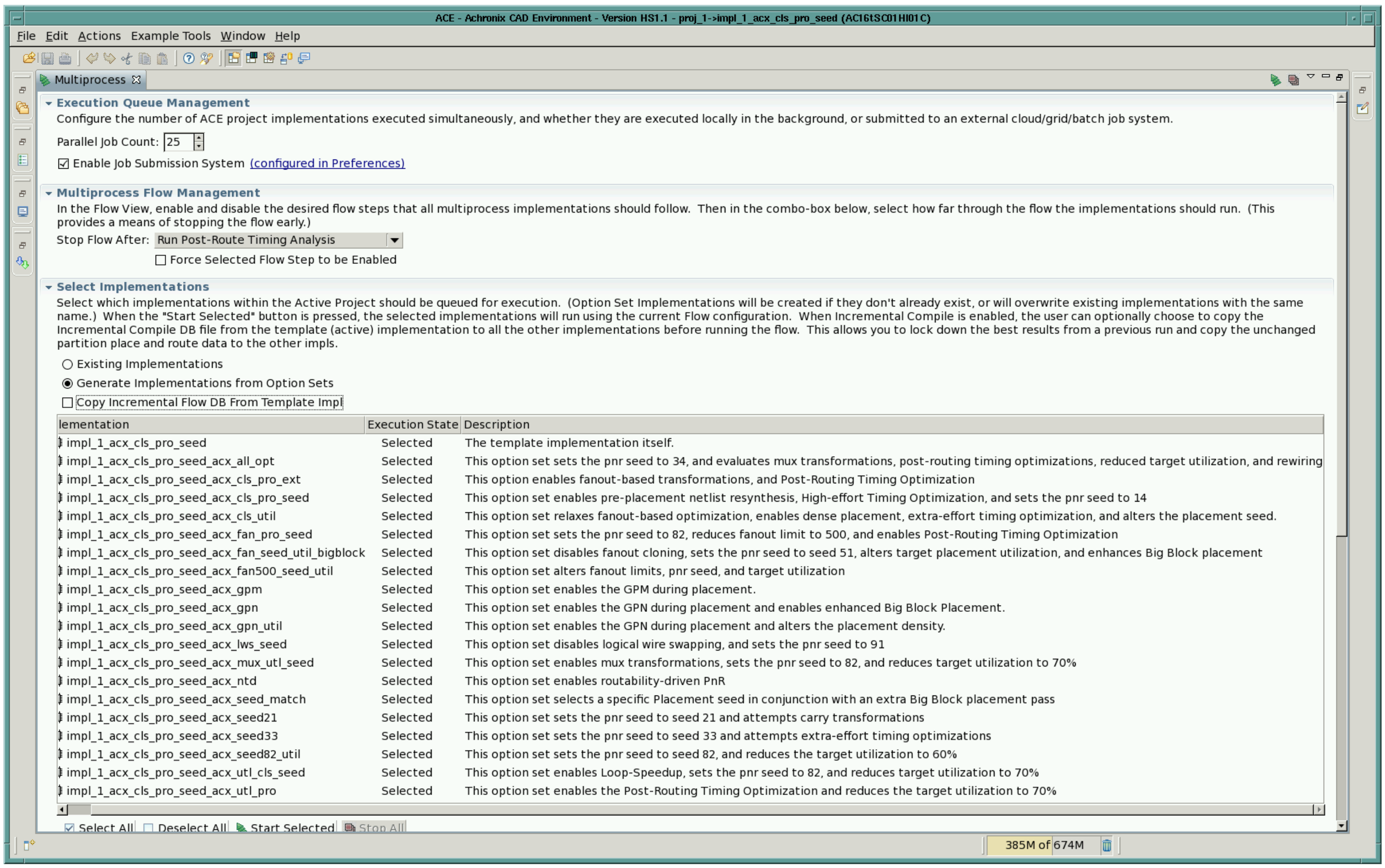
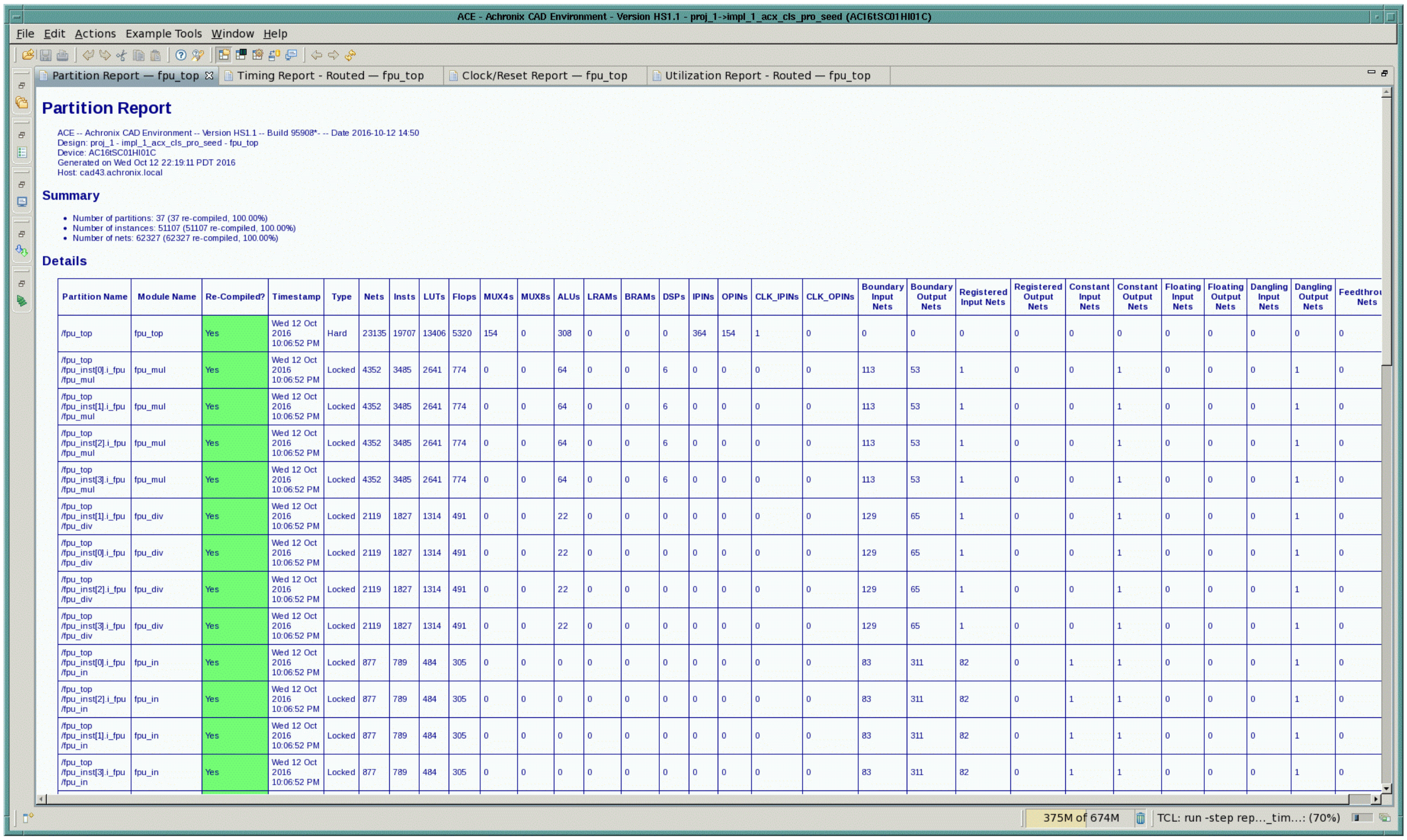


Figure 197: ***Multiprocess View Menus***

As in the single-process incremental compile flow, during the first pass through ACE, all implementations have their partitions compiled from scratch, as seen in the following screenshot of the partition report from one of the completed implementations.



Partition Report

ACE -- Achronix CAD Environment -- Version HS1.1 -- Build 95908*- -- Date 2016-10-12 14:50
Design: proj_1 - impl_1_acx_cls_pro_seed - fpu_top
Device: AC16tSC01HI01C
Generated on Wed Oct 12 22:19:11 PDT 2016
Host: cad43.achronix.local

Summary

- Number of partitions: 37 (37 re-compiled, 100.00%)
- Number of instances: 51107 (51107 re-compiled, 100.00%)
- Number of nets: 62327 (62327 re-compiled, 100.00%)

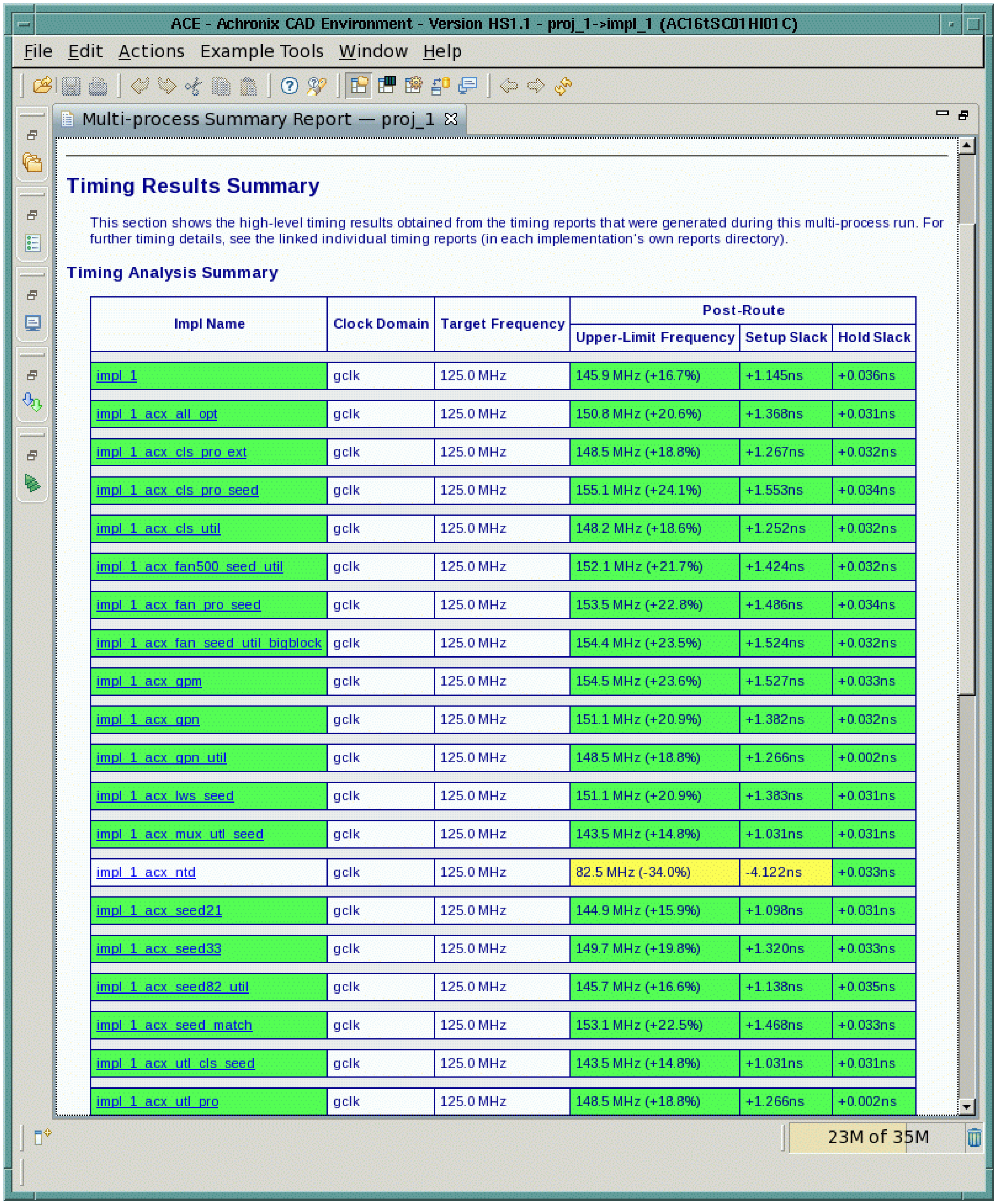
Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthrou Nets
/fpu_top	fpu_top	Yes	Wed 12 Oct 2016 10:06:52 PM	Hard	23135	19707	13406	5320	154	0	308	0	0	0	364	154	1	0	0	0	0	0	0	0	0	0	0	0	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[1].i_fpu /fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[2].i_fpu /fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[3].i_fpu /fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[0].i_fpu /fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
/fpu_top /fpu_inst[2].i_fpu /fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
/fpu_top /fpu_inst[1].i_fpu /fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
/fpu_top /fpu_inst[3].i_fpu /fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0

Figure 198: ***Partition Report from First Completed Multiprocess Implementation***

Step 3: Select the Implementation with Best Performance

After all of the parallel runs have completed, select the implementation with the best performance on the most timing-critical clock domain. A summary of the frequency, setup slack, and hold slack for each implementation is provided in the Multiprocess Summary Report (see the following screenshot).



Impl Name	Clock Domain	Target Frequency	Post-Route		
			Upper-Limit Frequency	Setup Slack	Hold Slack
impl_1	gclk	125.0 MHz	145.9 MHz (+16.7%)	+1.145ns	+0.036ns
impl_1_acx_all_opt	gclk	125.0 MHz	150.8 MHz (+20.6%)	+1.368ns	+0.031ns
impl_1_acx_cls_pro_ext	gclk	125.0 MHz	148.5 MHz (+18.8%)	+1.267ns	+0.032ns
impl_1_acx_cls_pro_seed	gclk	125.0 MHz	155.1 MHz (+24.1%)	+1.553ns	+0.034ns
impl_1_acx_cls_util	gclk	125.0 MHz	148.2 MHz (+18.6%)	+1.252ns	+0.032ns
impl_1_acx_fan500_seed_util	gclk	125.0 MHz	152.1 MHz (+21.7%)	+1.424ns	+0.032ns
impl_1_acx_fan_pro_seed	gclk	125.0 MHz	153.5 MHz (+22.8%)	+1.486ns	+0.034ns
impl_1_acx_fan_seed_util_bigblock	gclk	125.0 MHz	154.4 MHz (+23.5%)	+1.524ns	+0.032ns
impl_1_acx_gpm	gclk	125.0 MHz	154.5 MHz (+23.6%)	+1.527ns	+0.033ns
impl_1_acx_gpn	gclk	125.0 MHz	151.1 MHz (+20.9%)	+1.382ns	+0.032ns
impl_1_acx_gpn_util	gclk	125.0 MHz	148.5 MHz (+18.8%)	+1.266ns	+0.002ns
impl_1_acx_lws_seed	gclk	125.0 MHz	151.1 MHz (+20.9%)	+1.383ns	+0.031ns
impl_1_acx_mux_utl_seed	gclk	125.0 MHz	143.5 MHz (+14.8%)	+1.031ns	+0.031ns
impl_1_acx_ntd	gclk	125.0 MHz	82.5 MHz (-34.0%)	-4.122ns	+0.033ns
impl_1_acx_seed21	gclk	125.0 MHz	144.9 MHz (+15.9%)	+1.098ns	+0.031ns
impl_1_acx_seed33	gclk	125.0 MHz	149.7 MHz (+19.8%)	+1.320ns	+0.033ns
impl_1_acx_seed82_util	gclk	125.0 MHz	145.7 MHz (+16.6%)	+1.138ns	+0.035ns
impl_1_acx_seed_match	gclk	125.0 MHz	153.1 MHz (+22.5%)	+1.468ns	+0.033ns
impl_1_acx_utl_cls_seed	gclk	125.0 MHz	143.5 MHz (+14.8%)	+1.031ns	+0.031ns
impl_1_acx_utl_pro	gclk	125.0 MHz	148.5 MHz (+18.8%)	+1.266ns	+0.002ns

Figure 199: ***Multiprocess Summary Report from the First Incremental Run***

The following screenshot shows the critical path in the best implementation of the run, `impl_1_acx_cls_pro_seed` (actual results may differ).

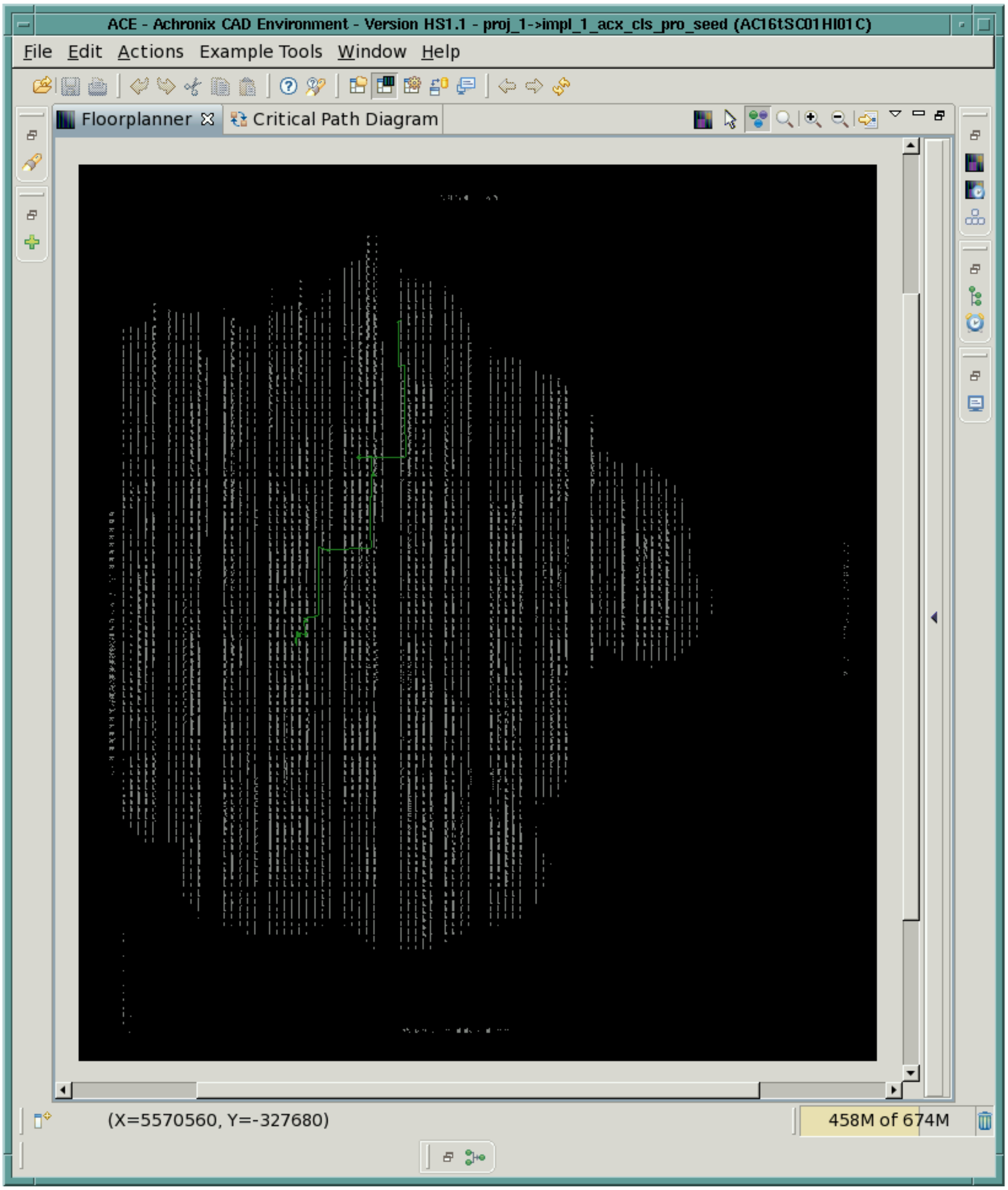


Figure 200: *Critical Path (green) in the Best Multiprocess Implementation*

Return to the Multiprocess View and check the **Copy Incremental Flow DB from Template Impl** checkbox. Optionally, change the radio button to **Existing Implementations** if desired (though this is not necessary). See the following screenshot.

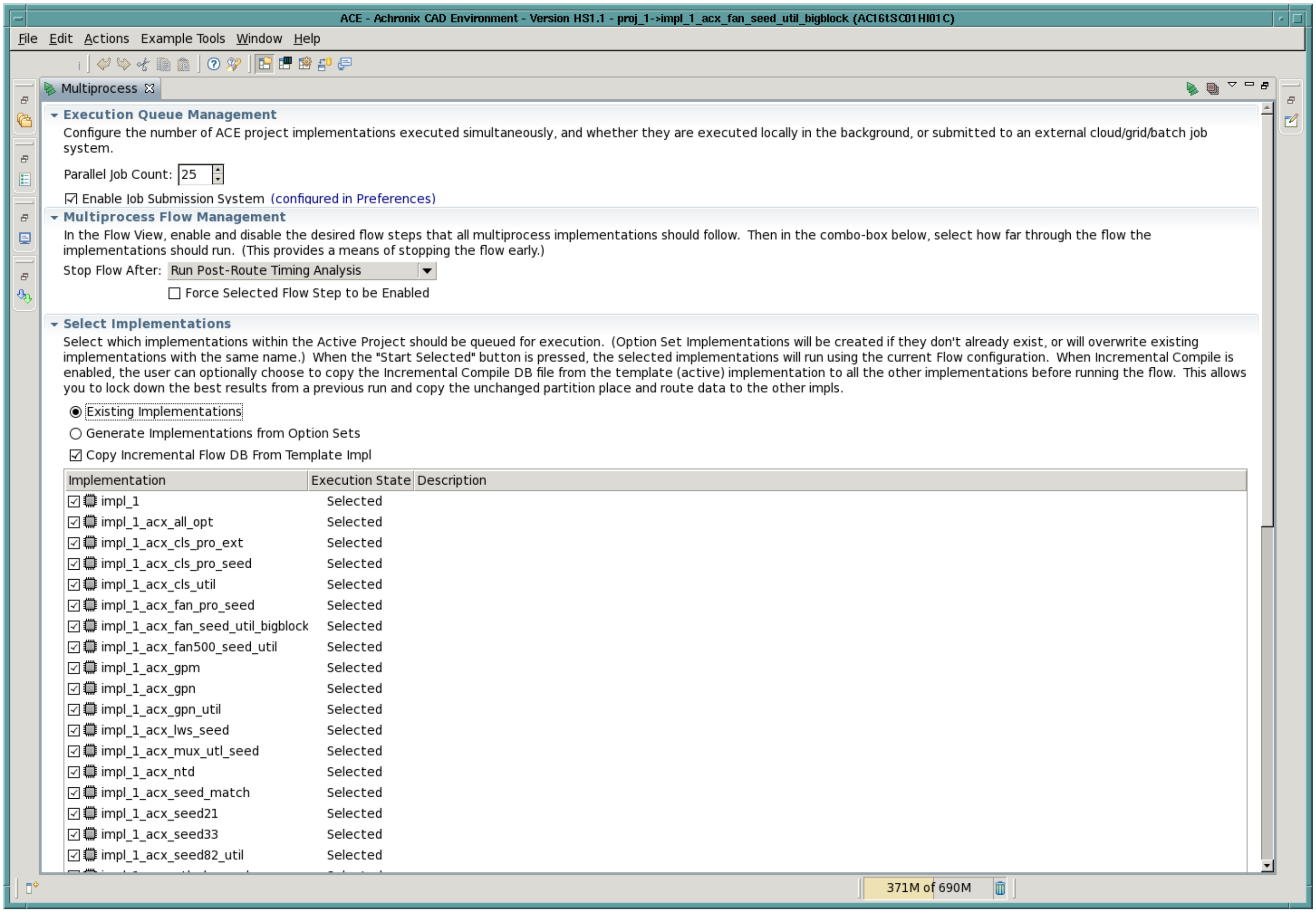


Figure 201: ***Multiprocess View with the Copy Incremental Flow DB from Template Impl Option Checked***

The Template implementation referred to in the checkbox is the implementation to be used as the source for all unchanged partitions in the next incremental compile. The Template implementation is the same as the Active implementation. From the Projects View of the Projects Perspective, click the triangle to the left of the Project name to expand the list of implementations. Then left-click on the desired implementation (the one with the best performance) to make it the Active implementation. The implementation name of the active implementation turns bold and is highlighted as in the following screenshot.

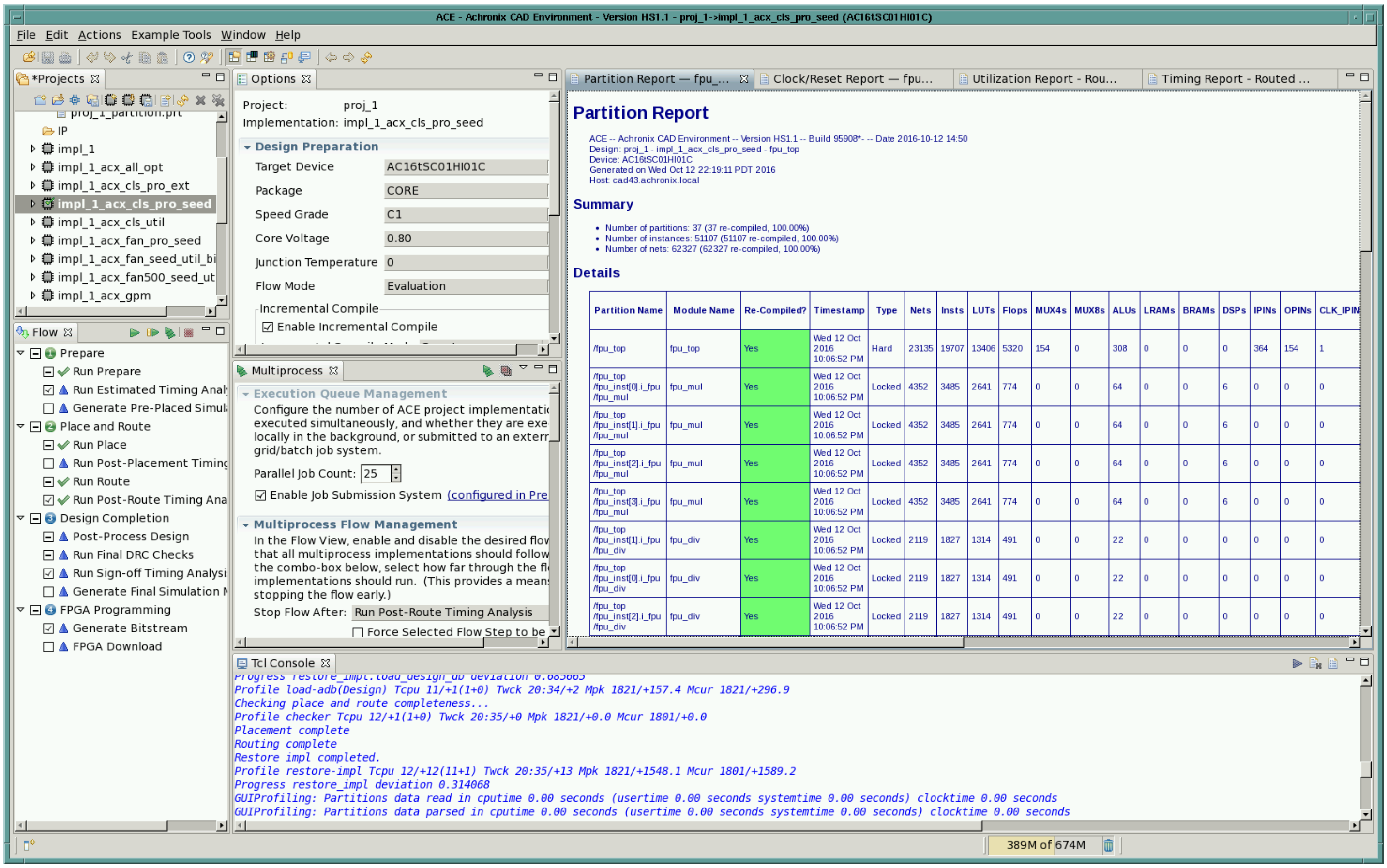


Figure 202: *Selection of the Template (Active) Implementation*

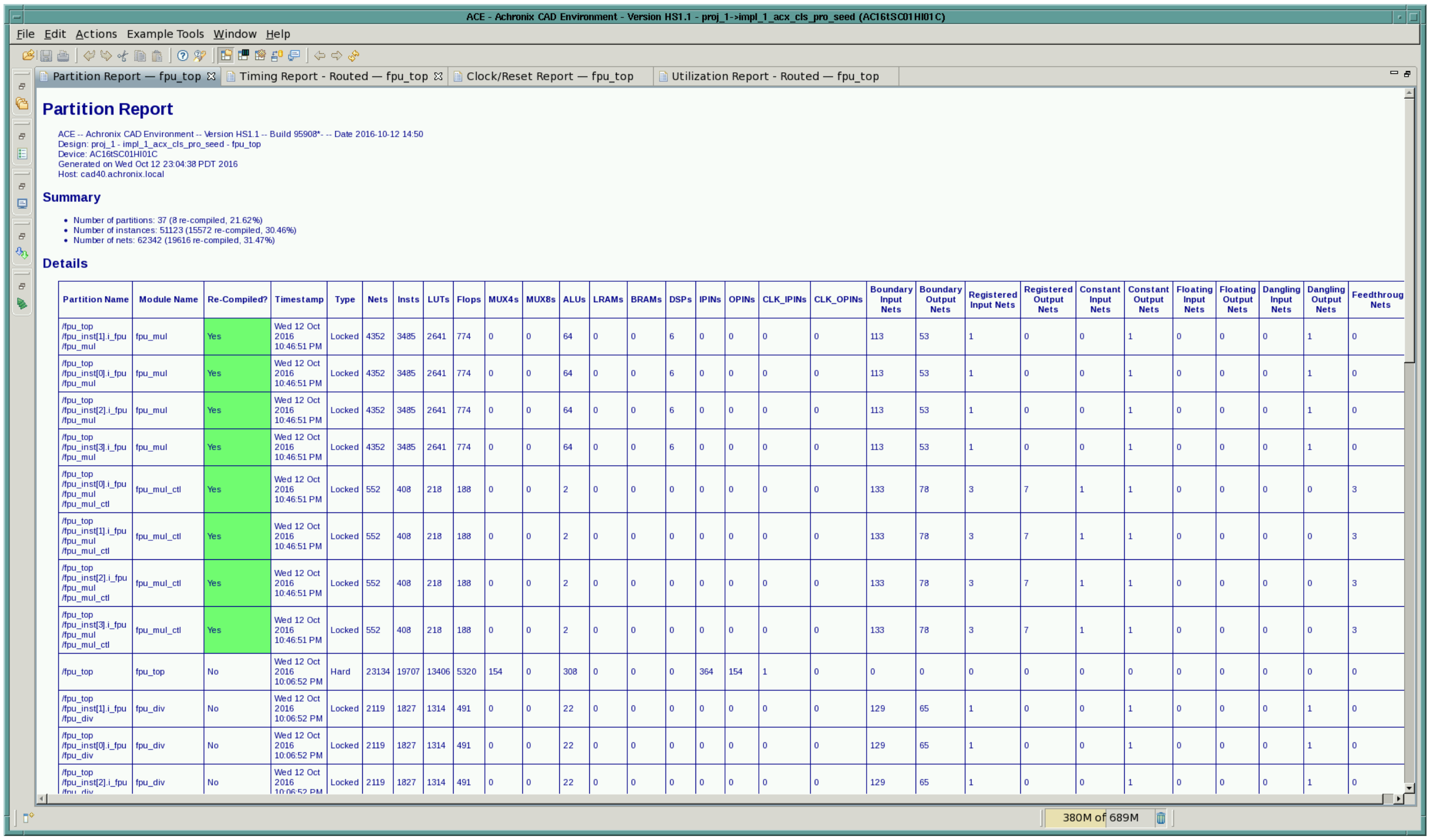
Step 4: Change the RTL and Recompile the Design in Synplify Pro

Repeat Steps 8-10 of the Single-Process Incremental Compile Tutorial (see page 367) to modify the RTL and force Syplify Pro to recompile the partitions in at least one of the defined compile points.

Step 5: Recompile the Multiprocess Implementations in ACE

Click the three stacked green triangle icon in the Multiprocess view to start a new incremental compile iteration on all implementations in parallel. ACE automatically copies the `output/<design>.icdb` file from the Template implementation into all other implementations and uses that as the source for the tear-and-stitch operation on all unchanged partitions during the run_prepare flow step.

As seen in the following screenshot from the `impl_1_acx_mux_utl_seed` implementation, only 8 of the 37 partitions have been recompiled in this iteration.



Partition Report

ACE -- Achronix CAD Environment -- Version HS1.1 -- Build 95908*- -- Date 2016-10-12 14:50
Design: proj_1 - impl_1_acx_cls_pro_seed - fpu_top
Device: AC16tSC01HI01C
Generated on Wed Oct 12 23:04:38 PDT 2016
Host: cad40.achronix.local

Summary

- Number of partitions: 37 (8 re-compiled, 21.62%)
- Number of instances: 51123 (15572 re-compiled, 30.46%)
- Number of nets: 62342 (19616 re-compiled, 31.47%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthroug Nets
/fpu_top /fpu_inst[1].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[2].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[3].i_fpu /fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[0].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
/fpu_top /fpu_inst[1].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
/fpu_top /fpu_inst[2].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
/fpu_top /fpu_inst[3].i_fpu /fpu_mul /fpu_mul_ctl	fpu_mul_ctl	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
/fpu_top	fpu_top	No	Wed 12 Oct 2016 10:06:52 PM	Hard	23134	19707	13406	5320	154	0	308	0	0	0	364	154	1	0	0	0	0	0	0	0	0	0	0	0	0
/fpu_top /fpu_inst[1].i_fpu /fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[0].i_fpu /fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
/fpu_top /fpu_inst[2].i_fpu /fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0

Figure 203: *Partition Report from Second Incremental Multiprocess Compile*

After all of the parallel runs complete, return to the updated Multiprocess Summary Report in the Multiprocess View to examine the critical path reports for each implementation.

As seen in the example below, the impl_1_acx_clk_pro_seed implementation achieved 155.1 MHz in the first iteration and 148.2 MHz in the second iteration. One of the incremental compiles, for impl_acx_fan_pro_seed, failed to route and returned a Flow Error.

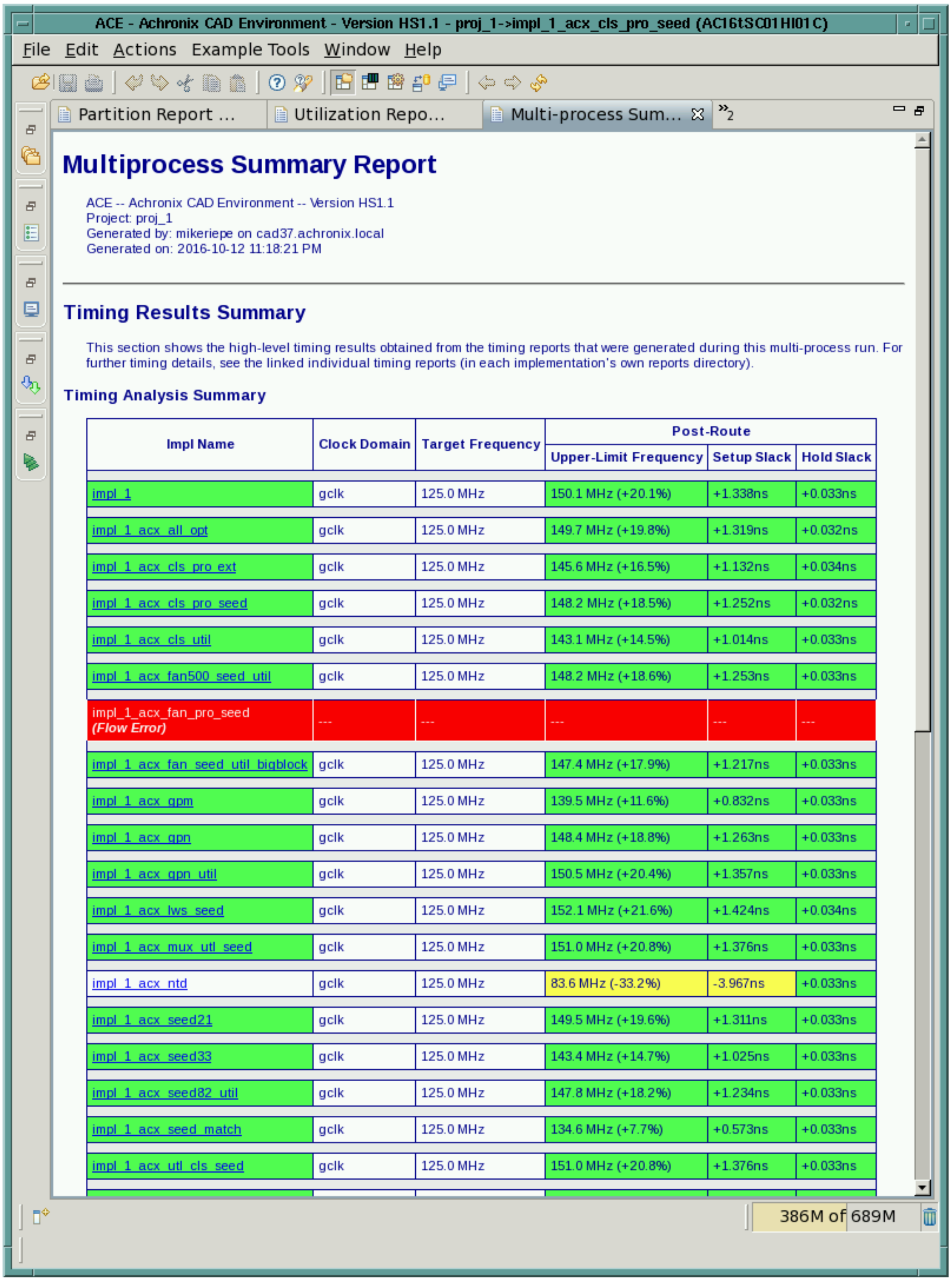


Figure 204: *Multiprocess Summary Report from the Second Incremental Run*

Finally, select **File→Restore Implementation** to restore the routed.acxdb database for the Template implementation, and select **Actions→Timing→Run Post-Route Timing Analysis** to observe the new critical path. The following screenshot shows the critical path in the Template implementation after recompiling all of the changed partitions. As usual, the instances of all unchanged partitions are highlighted in a dark yellow color to indicate that they are locked.

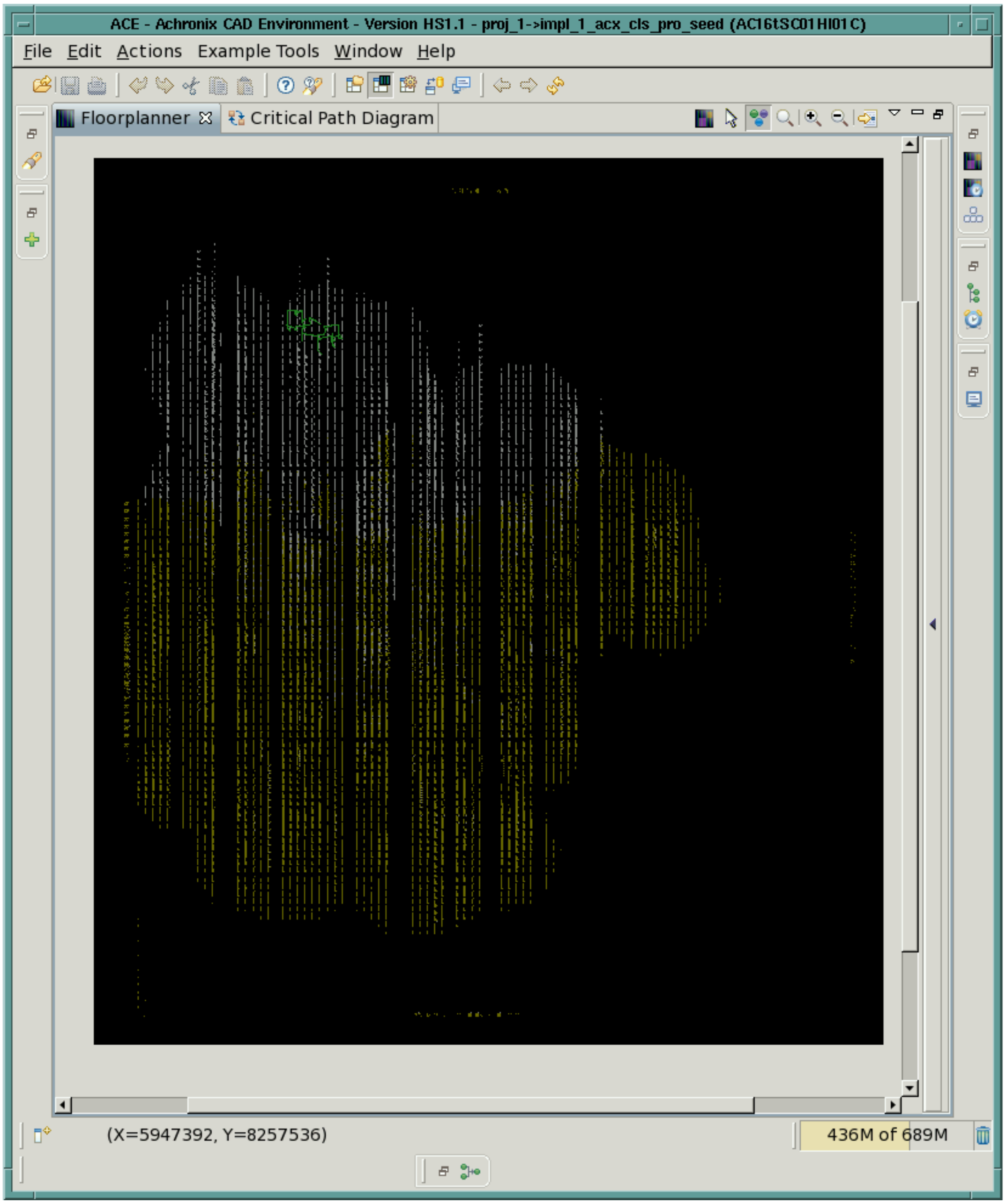


Figure 205: *Critical Path (Green) from the Best Multiprocess Implementation of the Second Incremental Run*

All timing paths inside the unchanged partitions can be observed to remain the same. Once timing closure of a critical block in at least one of the multiprocess implementations is achieved, this flow allows timing closure to be maintained until that block must be recompiled.

Automatic Flop Pushing into I/O Pads

The term “flop pushing” refers to the process of converting an unregistered pad and one or more attached DFFs into a registered pad for Speedcore devices. ACE performs this operation in the reconditioner during the `run_prepare` step. The purpose of this operation is to help with chip I/O timing closure. By avoiding the pad-to-flop, or flop-to-pad, delays, extra margin is achieved for off-chip timing paths.

Background

The flop-pushing feature is necessary in ACE because Synplify does not support inferencing of registered pads. The following Verilog source code describes a simple design consisting of black-box I/O pads, IPIN and OPIN instances, and two flip-flops. All three levels of a typical eFPGA design hierarchy are shown.

```
module DTM_TEST (in, clk, rst, ce, out)
    input in, rst, ce, clk;
    output out;

    wire in_d, rst_d, ce_d, clk_d, out_d;

    BB_PAD_IN  ipad_in  (.padin(in),   .dout(in_d) );
    BB_PAD_IN  ipad_rst (.padin(rst),  .dout(rst_d));
    BB_PAD_IN  ipad_ce  (.padin(ce),   .dout(ce_d) );
    BB_PAD_CLK ipad_clk (.padin(clk),  .dout(clk_d));
    BB_PAD_OUT opad_out (.padout(out), .din(out_d) );

    STM_TEST stm_test (.in(in_d), .rst(rst_d), .ce(ce_d), .clk(clk_d), .out(out_d));

endmodule

module STM_TEST (in, rst, ce, clk, out)
    input in, rst, ce, clk;
    output out;

    IPIN     ipin_in  (.din(in),  .dout(in_p) );
    IPIN     ipin_rst (.din(rst), .dout(rst_p));
    IPIN      ipin_ce  (.din(ce),  .dout(ce_p) );
    CLK_IPIN ipin_clk (.din(clk), .dout(clk_p));
    OPIN     opin_out (.din(out), .dout(out_p));

    UCM_TEST ucm_test (.in(in_p), .rst(rst_p), .ce(ce_p), .clk(clk_p), .out(out_p));

endmodule

module UCM_TEST (in, rst, ce, clk, out)
    input in, rst, ce, clk;
    output out;
    reg out;

    wire dff_q;

    ACX_DFFER dff (.d(in), .clk(clk), .ce(ce), .rn(rst), .q(dff_q));

    always @(posedge clk or negedge rst)
    begin
        if (!rst)
            out <= 0;
        else
            if (ce) out <= dff_q;
    end
endmodule
```

Since the DFFER instance is driven directly by an input port, and the behavioral flip-flop directly drives an output port, one might expect RTL synthesis to generate registered input and output pads (with reset and enable). However, Synplify Pro generates an input pad, an output pad, and separate DFFER instances.

By placing the flops in the device core as separate instances, extra delay from the pad to the flop is required by the ring-to-core routing path. If the device I/O timing is tight, that could result in a setup timing failure. On the other hand, the presence of separate DFFER instances allows the flops to be placed in the core near their fanin/fanout logic, possibly reducing the routing delay to intermediate logic in the design. Flop pushing can therefore be viewed as a form of retiming, allowing the designer the ability to trade off-chip for on-chip delays between the pad and the flop.

Capabilities

In ACE, flop pushing is performed by the reconditioner during the run_prepare flow step. Flop pushing happens very early in the flow, after flattening but before I/O elaboration so that the reconditioner can operate directly on the IPAD and OPAD instances before they are elaborated into networks of separate `io_buffers` and datapath instances in the traditional Speedster FPGA flow. After elaboration, ACE would require detailed knowledge of the IPAD and OPAD implementations to convert an unregistered pad into a registered pad. For a Speedcore eFPGA instance, there are no PAD instances. Instead, there are IPIN and OPIN instances. Flop pushing in ACE operates on Speedcore IPIN and OPIN instances in the same way it operates on IPAD and OPAD instances in a traditional Speedster FPGA.

In the simplest case, ACE performs flop pushing by:

1. Finding an IPIN that drives a DFF, or a DFF that drives an OPIN
2. Deleting the DFF
3. Converting the IPIN/OPIN into a flopped IPIN/OPIN (by setting the "mode" parameter), connecting the DFF clock input to the IPIN/OPIN clock input, and optionally connecting the DFF reset and enable inputs to the IPIN/OPIN reset and enable inputs.

Flop pushing is supported for flops connected to IPIN data output pins, and OPIN data input pins. ACE also supports more complex cases:

- IPINs that drive more than one DFF
- Chains of buffer LUTs and inverter LUTs between the pad and the DFF
- OPINs in which the data input and the clock-enable input are both driven by DFFs

The reconditioner checks for many possible scenarios that prevent flops pushing, especially in the above complex cases. A partial list includes:

- The IPIN or OPIN is already registered
- DFFs in the IPIN fanout do not all share the same clock input nets
- DFFs in the IPIN fanout do not all share the same set, reset, or enable input nets
- DFFs in the IPIN fanout are a mixture of DFF, DFFC, DFFP, DFFR, and/or DFFS instances
- DFFs in the IPIN fanout have a mixture of synchronous and asychronous resets
- DFFs in the IPIN fanout are a mixture of positive/negative edge triggered
- DFFs in the IPIN fanout have different `init` parameter values
- A DFF in the IPIN fanout is driven by more than one input pad, or drives more than one output pad
- A DFF clock is driven by a generated clock or reset that can only be routed in the core
- LUTs between the IPIN/OPIN and DFF are configured as anything but a buffer or inverter
- Nets on the path between the IPIN/OPIN and DFF (including intermediate buffers or inverters) have a fanout greater than one
- DFFs driven by a IPIN/OPIN through a chain of buffers and/or inverters that have different inversion (odd vs. even number of inverters)

ACE Attributes

The behavior of ACE with respect to flop pushing can be controlled for individual input/output pads by attributes in the RTL or PDC constraints.

Depending on the value of the `push_flops_into_pads` implementation option (see Implementation Options (see page 424)), ACE looks for an attribute named `syn_useioff` with the following semantics:

- If the syn_useioff attribute associated with an I/O pin has a non-zero value, ACE pushes flip-flops into that pin when possible. This behavior is useful when the push_flops_into_pads implementation option has the value 1 (manual mode).
- If the syn_useioff attribute associated with an I/O pin has the value 0, ACE prevents flip-flops from being automatically pushed into that pin. This behavior is useful when the push_flops_into_pads implementation option has a value of 15 (automatic mode).

The syn_useioff attribute can be placed in several different locations in your RTL code, as described and shown in the following.

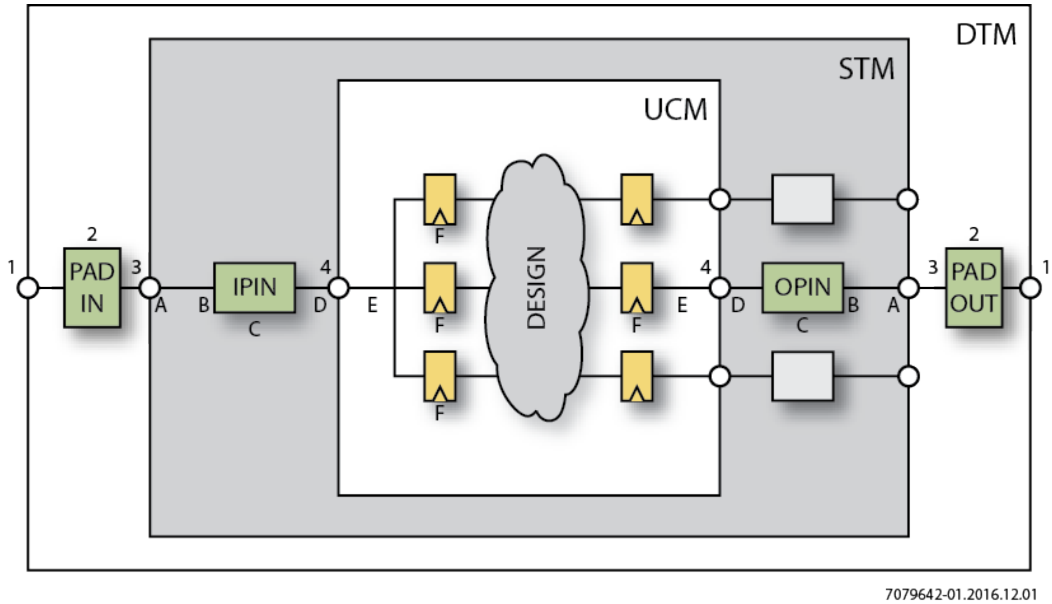


Figure 206: ***Valid Locations to Place the syn_useioff Attribute in an eFPGA Design Hierarchy***

- On an STM port (A) connected to an IPIN/OPIN (but only if the STM is targeted as the top-level module in ACE)
- On the net (B) connecting an IPIN/OPIN to an STM port
- On an IPIN/OPIN instance (C)
- On a net (D) connecting an IPIN/OPIN to a UCM port
- On a net (E) connecting a UCM port to one or more flip-flop instances
- On the flip-flop instances (F) driven by an IPIN instances or driving an OPIN instance

If there is more than one flip-flop driven by the same IPIN, all DFF instances must have a syn_useioff attribute with the same value. Specifically, the syn_useioff *cannot* be placed in the following locations:

- On a DTM port (1) connected to a black-box instance or directly driving the STM port because ACE does not trace through the DTM black-box network searching for the syn_useioff attributes
- On a black-box instance (2)
- On the net (3) connecting an STM pin to the black-box network in the DTM
- On the UCM port (4) separating the STM and the UCM hierarchies because Synplify does forward annotate attributes on intermediate output ports, but when placed on intermediate input ports they are lost during flattening, so this method is not recommended

Placing the `syn_useioff` attribute directly on the top-level module ports (A) is useful in Speedster designs, where the top-level ports directly connect to the I/O pads. That technique can also be used for Speedcore eFPGA designs when the STM level of hierarchy is targeted as the top-level module in ACE and can be user modified. For Speedcore designs in which the STM level of hierarchy is fixed by the ASIC integrator, and only the UCM hierarchy can be user modified, the most convenient locations to place the `syn_useioff` attribute may be the UCM port-to-core nets (E) or the DFF instances (F).

Be careful not to add the `syn_useioff` attributes in more than one location. A conflict occurs when some attributes associated with particular I/O pad or pin have the value "1", and some have the value "0". When a conflict is detected, ACE issues a warning message, the value "0" is assumed, and flop pushing is disabled for that I/O. Be especially careful to avoid conflicts when the attributes are placed on DFF instances, and an IPIN drives more than one DFF.

The `syn_useioff` attributes can be specified by the user in the Verilog/VHDL source code, or in the physical design constraints (`.pdc`) file, as follows. An advantage of specifying the attribute in the `.pdc` file is that it is not necessary to re-synthesize the design in order to experiment with different flop-pushing strategies. An advantage of specifying the attribute in the HDL source code is that the intent of the designer is more self-documenting, as readers do not have to refer to a separate `.pdc` file and cross-reference the port/net/instance names between the files.

Examples

Several examples follow demonstrating how to set the `syn_useioff` attribute on:

- top-level ports
- I/O pin instances
- boundary wires
- DFF instances

also demonstrated is whether to use either Verilog, VHDL, or a PDC constraint. For more information about the use of attributes in Synplify see the section "Forward Annotation of RTL Attributes to Netlist" in the Synthesis Optimization Recommendations chapter of the *Synthesis User Guide* (UG018).

Verilog Example of a Port attribute

```
module flop_push_test1 (ina, inb, sel, clk, z0);
    input [3:0] ina /* synthesis syn_useioff=1 */;
    input [3:0] inb /* synthesis syn_useioff=0 */;
    input sel       /* synthesis syn_useioff=1 */;
    input clk;
    output z0       /* synthesis syn_useioff=1 */;
endmodule
```

VHDL Example of a Port Attribute

```
entity flop_push_test1 is
port(
        ina    : in signed( 3 downto 0 );
        inb    : in signed( 3 downto 0);
        sel    : in std_logic;
        clk    : in std_logic;
        z0     : out std_logic
);

attribute syn_useioff : boolean;
attribute syn_useioff of ina : signal is TRUE;
attribute syn_useioff of inb : signal is FALSE;
```

```
attribute syn_useioff of sel : signal is TRUE;
attribute syn_useioff of z0 : signal is TRUE;

end entity;
```

```
Physical Design Constraints (.pdc) File of Port Attributes

set_property syn_useioff "1" [find -ports {ina\[*\]}]
set_property syn_useioff "0" [find -ports {inb\[*\]}]
set_property syn_useioff "1" {p:sel p:z0}
```

> **Note**
>
> In the above three examples, the input PortBus `ina`, the input port `sel`, and the output port `z0` are selected for flop pushing. The input PortBus `inb`, the input port `clk`, are not.
>
> If the attribute has the value "1", ACE tries to push a flip-flop into the pad connected to the given port even when flop pushing is disabled by default. If the attribute has the value "0", ACE prevents flop pushing on that pad, even if flop pushing is enabled by default. Of course flop pushing may be prevented by any of the exceptions listed above (see page 419).
>
> It is not possible, using RTL attributes, to apply different values of the `syn_useioff` attribute to different ports in a port bus (e.g, giving `in_a[2]` a `syn_useioff` value of "0"). It cannot even be done by applying an attribute to a wire that is assigned the value of the bus (see the following example, `flop_push_test5`). The solution requires bit-blasting the bus into separate ports or assigning different values of `syn_useioff` to different bus ports using PDC.

```
Verilog Example of an IPIN Instance Attribute

module flop_push_test2 (in, clk);
   input [38:0] in;
   input clk;

   wire ipad_dout_37;
   BB_IPAD ipad_37 (.pad(in[37]), .dout(ipad_dout_37) );
   wire ipin_dout_37;
   IPIN ipin_37( .din(ipad_dout_37) , .dout(ipin_dout_37) ) /* synthesis syn_useioff = 0 */;

   reg data_37 = 1'b0;
   always @(posedge clk)
     begin
        data_37 <= ipin_dout_37;
     end

endmodule
```

```
Physical Design Constraints (.pdc) File of IPIN Instance Attributes

set_property syn_useioff "1" [find -insts {ipin_*}]
set_property syn_useioff "0" {i:ipin_37}
```

```
Verilog Example of a Boundary Wire Attribute

module flop_push_test3 (in, clk);
```

```
   input [38:0] in;
   input clk;

   (* syn_keep *) wire ipin_dout_37 /* synthesis syn_useioff=0 */;
   IPIN ipin_37 (.pad(in[37]), .dout(ipin_dout_37));

   reg level1_37 = 1'b0;
   always @(posedge clk[0])
     begin
        level1_37 <= ipad_dout_37;
     end
endmodule
```

Physical Design Constraints (.pdc) File of Wire Attributes

```
set_property syn_useioff "1" [find -nets {ipin_dout_*}]
set_property syn_useioff "0" {n:ipin_dout_37}
```

Verilog Example of a DFF Instance Attribute

```
module flop_push_test4 (in, clk);
    input [38:0] in;
    input clk;

    wire ipad_dout_37;
    BB_IPAD ipad_37 (.pad(in[37]), .dout(ipad_dout_37));

    wire ipin_dout_37;
    IPIN ipin_37(.din(ipad_dout_37), .dout(ipin_dout_37) );

    wire dff1_q, dff2_q;
    ACX_DFF dff1 (.d(ipin_dout_37), .clk(clk), q(dff1_q))  /* synthesis syn_useioff = 0 */;
    ACX_DFF dff2 (.d(ipin_dout_37), .clk(clk), q(dff2_q))  /* synthesis syn_useioff = 0 */;
endmodule
```

Physical Design Constraints (.pdc) File of DFF Instance Attributes

```
set_property syn_useioff "1" [find -insts {dff*}]
set_property syn_useioff "0" {i:dff1 i:dff2}
```

The following is an example of a boundary wire attribute that *does not* work as expected.

Verilog Example of a Boundary Wire Attribute that DOES NOT work

```
module flop_push_test5 (in, clk);
    input [38:0] in;
    input clk;

    (* syn_keep *) wire ipin_din_37 /* synthesis syn_useioff=0 */;
    assign ipin_din_37 = in[37];

    wire ipin_dout_37;
    IPIN ipin_37 (.pad(ipin_din_37) , .dout(ipin_dout_37) );

    reg data_37 = 1'b0;
```

```
    always @(posedge clk[0])
        begin
            data_37 <= ipad_dout_37;
        end
endmodule
```

Caution!

As noted above, it is not possible using RTL attributes to apply different values of the `syn_useioff` attribute to different ports in a port bus. The above example, `flop_push_test5`, appears to be a clever way to apply the `syn_useioff` attribute to the net connecting the UCM port to a single IPIN in the 39-bit port bus `in`. Unfortunately, this technique does not work. Synplify Pro optimizes away the wire `ipin_din_37`, despite the presence of the `syn_keep` attribute, and the `syn_useioff` attribute is not forward-annotated into the ACE input netlist. One can use PDC, however, to apply the `syn_useioff` attribute to the IPIN instance as in the following example.

```
Physical Design Constraints (.pdc) For Example flop_push_test5 That DOES Work

set_property syn_useioff "1" [find -nets {in\[*\]}]
set_property syn_useioff "0" {n:in\[37\]}
```

Implementation Options

The behavior of ACE with respect to flop pushing is controlled by the implementation options `push_flops_into_pads` and `pad_flop_pushing_clock_type`, described in the following section. Also, see Options View (see page 103).

push_flops_into_pads

The implementation option, `push_flops_into_pads`, controls whether flop pushing is performed automatically or manually. This implementation option has the following legal settings:

- "0" – flop pushing is completely disabled
- "1" – (manual mode) push flops into pads that have the `syn_useioff` attribute set to "1"
- "15" – (automatic mode) push flops into all pads *except* those that have the `syn_useioff` attribute set to "0"

```
Example Setting

set_impl_option push_flops_into_pads 15
```

pad_flop_pushing_clock_type

The implementation option, `pad_flop_pushing_clock_type`, enables automatic flop pushing to be controlled by the routing type of the pushed clock. The option only applies when the implementation option `push_flops_into_pads` has the value "15" (automatic mode). This implementation option has the following legal settings:

- "boundary" – automatically push flops into pads only when the flops are clocked by a boundary clock
- "trunk" – automatically push flops into pads only when the flops are clocked by a trunk clock
- "all" – automatically push flops into all pads regardless of the clock routing type

The routing type of a clock net is controlled by the `set_clock_type` command.

```
set_clk_type Examples

set_clock_type clk1 -boundary
set_clock_type clk2 -trunk
set_impl_option pad_flop_pushing_clock_type "boundary"
```

In the above example, only flops clocked by the boundary clock `clk1` are automatically pushed in the pads.

Timing Analysis Implications

As discussed above, flop pushing can be viewed as a form of retiming. Pushing a flop into a boundary pin reduces the off-chip timing path at the flop input by reducing the wiring delay. But it increases the on-chip timing path at the flop output, possibly by a large amount depending on how closely the driven logic is placed to the boundary pin. On small designs, especially when the logic is placed near the center of the chip, the increased delay can be significant.

Enabling flop pushing by default across a suite of designs often causes QoR to degrade significantly. This degradation happens because the off-chip delays are often not modeled well in the design timing constraints. The off-chip delay must be modeled using a `set_input_delay` or `set_output_delay` timing constraint. If those delays are zero, the improvement in off-chip delay may not be evident, and the increase in on-chip delay may be dominant.

More commonly, constraints are not even given, causing the timer to completely ignore timing paths that start or end off-chip. Pushing a flop into a pad with unspecified input/output delay could cause new setup/hold violations to appear that were not previously modeled.

Working with Virtual I/O

The role of I/O virtualization is to take a design with too many I/O pads (or boundary pins in the case of a Speedcore fabric) and reduce the number of I/Os until the design fits in the given fabric. This option is only run in evaluation flow mode.

Behavior

I/O virtualization is performed automatically as part of the `run_prepare` flow step in evaluation flow mode. It is not permitted to export a bitstream for a design with virtualized I/Os. If there are a sufficient number of boundary pin sites to place the design, then the command is a no-op. If the design has more boundary pin instances than available sites, I/O virtualization modifies the netlist by reducing the number of boundary pins until the design fits. When virtualizing I/O, no attempt is made to maintain logical equivalency with the original netlist. Rather, the goal is to perturb the behavior of the placement and routing tools as little as possible and, therefore, make an evaluation run correspond as closely as possible to a production run of a similar design.

I/O virtualization operates by collapsing multi-bit bused boundary pins as well as single non-bused boundary pins. By default, pins are selected automatically for virtualization, starting with the widest pin bus until enough boundary pin sites are available to fit the remaining number of boundary pin instances. If that number is insufficient, individual non-bused pins are also virtualized. Pin buses and individual pins can also be manually selected for virtualization through top-level port attributes in the RTL or PDC constraints (see Port Attributes (see page 427), below). Depending on the virtualization mode, some serialization boundary pins may be inserted, so it is possible for the process to fail and leave too many boundary pins in the design.

The pins are collapsed using one of three user-specified styles:

- **stubout** – each IPIN is replaced with a "stub" LUT that drives a constant zero onto the IPIN output net. Similarly, each OPIN is replaced with a "stub" LUT, driven by the OPIN input net, with a floating output pin. These stub LUTs are given `must_keep` attributes so that ACE does not optimize them away.

- **serialize_dff** – bused IPINs are replaced with a single IPIN that drives a scan chain implemented with DFFs. The output of each DFF drives the output net from its original corresponding IPIN as well as the next stage in the scan chain. Bused OPINs are replaced with a single OPIN that is driven by a scan chain implemented with DFFs. The input of each DFF is driven by a 2-input MUX, one input of which is driven by the input net from its original corresponding OPIN. The other input of the MUX is driven by the output of the DFF in the previous stage of the scan chain. One additional IPIN per port bus is also added which drives the select line of the MUXes.
- **serialize_lut** – this style is the same as the `serialize_dff` style, except that the scan chain is implemented with LUTs instead of DFFs.

The stubout style is the simplest and has the greatest rate of pad compression. However, it is the least realistic since there are no connections pulling the stub LUTs toward the edge of the chip. The placer pulls them into the chip core, placing them at the center of gravity of the loads that they drive. The two serialize styles keep one representative boundary pin and are, therefore, more realistic, though of course the strength of the placement forces pulling the scan chain toward to chip edge are significantly reduced. The main reason to use the `serialize_dff` style over the `serialize_lut` style is that, in the former style, the timer only sees a path that starts or ends at the last stage of the scan chain, while in the former the entire scan chain (possibly hundreds of LUT delays) contributes to the length of the timing path.

For the `serialize_dff` style, a clock must be connected to the DFFs that make up the scan chain. By default, that clock is selected automatically to be the clock in the chip core with the largest number of load pins. The clock can be user-specified either through a global implementation option, or on a per-port basis through an attribute on the port. These options are discussed in Implementation Options (see page 426) and Port Attributes (see page 427).

Implementation Options

The behavior of I/O virtualization can be controlled on a global basis by the following implementation options:

- **virtual_io_style** – controls the style, or method, used to virtualize excess I/O pad or boundary pin buses in the top-level netlist. Legal enumeration values are:
  - `stubout` (the default)
  - `serialize_dff`
  - `serialize_lut`

  See above for a definition of the behavior of each of these styles.
- **virtual_io_utilization** – sets the I/O pad or boundary pin utilization percentage targeted by I/O virtualization. Legal values are integers between `0` and `100`. An error is returned if the given utilization cannot be met. A target utilization of zero percent requests that all possible port buses and non-bused ports are to be virtualized to achieve the smallest possible number of pins. A target utilization of 100 percent requests that port busses and non-bussed ports are to be virtualized until the number of remaining ports fit into the target fabric. This option is mutually exclusive with the `virtual_io_num_pads` option (both cannot be specified).
- **virtual_io_num_pads** – sets the final number of I/O pad or boundary pin instances targeted by I/O virtualization. Legal values are `0` or larger. A target pad number of zero requests that all possible port buses and non-bused ports are to be virtualized to achieve the smallest possible number of pins. If the specified value is larger than the number of available I/O pad or boundary pin sites in the selected fabric, the number of available I/O pad or boundary pin sites are targeted. This option is mutually exclusive with the `virtual_io_utilization` option (both cannot be specified).
- **virtual_io_clock_port** – specifies the name of the clock, by its top-level port name, to be used by I/O virtualization to clock serialization flops. Only applies for the `serialize_dff` virtualization style. This option can also be specified individually for a given port with the RTL or PDC port attribute, `ace_virtualize_clock_port`, which overrides this option if given. If not specified, the virtualization clock is derived automatically as the core clock net driving the largest number of loads. This option is mutually exclusive with the `virtual_io_clock_net` option (both cannot be specified).

- **virtual_io_clock_net** – specifies the name of the clock, by its net name, to be used by I/O virtualization to clock serialization flops. Only applies for the `serialize_dff` virtualization style. This option can also be specified individually for a given port with the RTL or PDC port attribute, `ace_virtualize_clock_net`, which overrides this option if given. If not specified, the virtualization clock is derived automatically as the core clock net driving the largest number of loads. This option is mutually exclusive with the `virtual_io_clock_port` option (both cannot be specified).

Port Attributes

By default, I/O virtualization selects port buses for virtualization automatically. They are virtualized in order of decreasing size until the netlist meets the given target boundary pin utilization. However, port buses which are virtualized through the use of the RTL port attribute `ace_virtualize` can be manually controlled. When the virtualization style is set to `serialize_dff`, either a top-level port name or net name to be connected to the clock input of the new serialization flop instances can also be specified. Use the RTL port attribute `ace_virtualize_clock_port` or `ace_virtualize_clock_net` respectively.

The attribute can be set in the Verilog/VHDL source code, or in the physical design constraints (`.pdc`) file, as follows. An advantage of setting the property in the `.pdc` file is that the design does not have to be re-synthesized in order to experiment with different virtualization strategies.

Verilog Example

```
module pds (
    input                                   clk_i,
    (* ace_virtualize="1", ace_virtualize_clock_port="clk_i" *)
    output  [63:0]                          tx_data_o,
    (* ace_virtualize="1", ace_virtualize_clock_net="clk_i_c" *)
    output  [ 7:0]                          tx_ifg_delay_o
endmodule
```

VHDL Example

```
entity pds is
port(
        clk_i          : in std_logic;
        tx_data_o      : out signed( 63 downto 0);
        tx_ifg_delay_o : out std_logic_vector(7 downto 0)
        );

attribute ace_virtualize : boolean;
attribute ace_virtualize of tx_data_o : signal is TRUE;
attribute ace_virtualize of tx_ifg_delay_o : signal is TRUE;

attribute ace_virtualize_clock_port : string;
attribute ace_virtualize_clock_net : string;
attribute ace_virtualize_clock_port of tx_data_o : signal is "clk_i";
attribute ace_virtualize_clock_net of tx_ifg_delay_o : signal is "clk_i_c";

end entity;
```

If the target boundary pin utilization is not met after all user-specified ports are virtualized, additional ports are selected automatically until the target boundary pin utilization is met.

```
Physical Design Constraints (.pdc) File

set_property ace_virtualize "1" [find -ports {sample_src\[*\]}]
set_property ace_virtualize_clock_port "clk" [find -ports {sample_src\[*\]}]
```

Runtime Messages

Below are the output messages from I/O virtualization using the example above with user-specified port buses and the `serialize_dff` virtualization style.

```
Runtime Messages

INFO: Virtualize IO: Serializing user-specified 512-bit output PortBus tx_data_o using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 64-bit output PortBus tx_data_valid_o using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 64-bit output PortBus tx_ifg_delay_o using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 512-bit input PortBus rx_data_i using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 128-bit output PortBus pause_val_o using clock clk_i_c
INFO: Virtualize IO: Serializing remaining 6 auto-selected input ports using clock clk_c
INFO: Virtualize IO: Serializing remaining 13 auto-selected output ports using clock clk_c

WARNING: Virtualize IO: Netlist pds had too many IOs to fit in the selected device. Merged and deleted
1280 of 1498 IO ports. Final number of ports is 227. This is for evaluation purposes only and will cause
simulation mismatches.
```

Schematic View

The following images illustrate each of the available virtualization styles with schematic diagrams showing 4-bit busses of input pads and output pads. First shown is the input netlist before pad virtualization, followed by the output netlist for the stubout, serialize_dff, and serialize_lut styles.

Input Netlist

The following figure illustrates the input netlist for a 4-bit bus of input pads, and a 4-bit bus of output pads. The output pads have an output-enable driver.

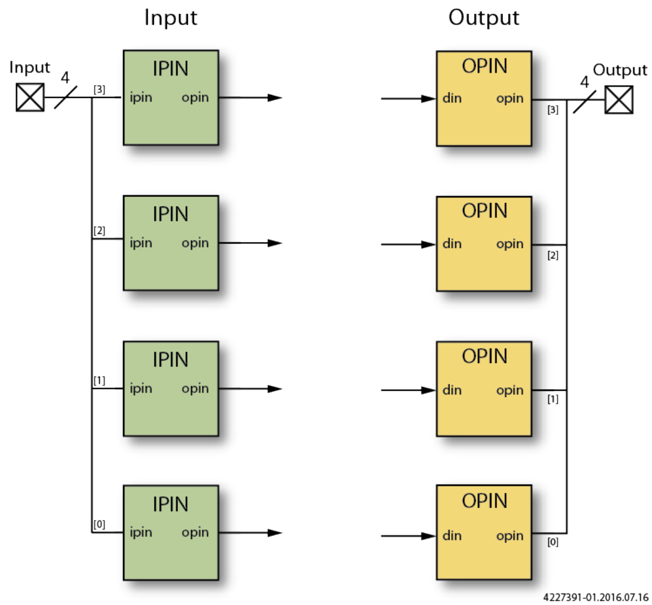


Figure 207: ***Input and Output Pads***

Output Netlist Styles

stubout

The following two schematics illustrate the output of I/O virtualization when using the stubout style. Notice that none of the IPIN or OPIN instances remain. The new LUTs replacing the IPIN instances are all driven by constant zeros, and the new LUTs replacing the OPIN instances have unconnected outputs.

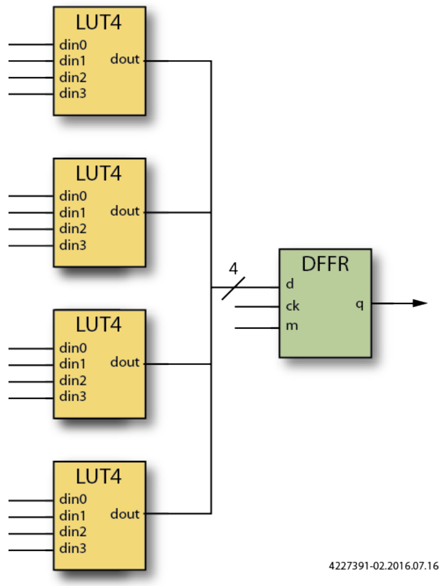


Figure 208: ***Stubout Style Input Pad***

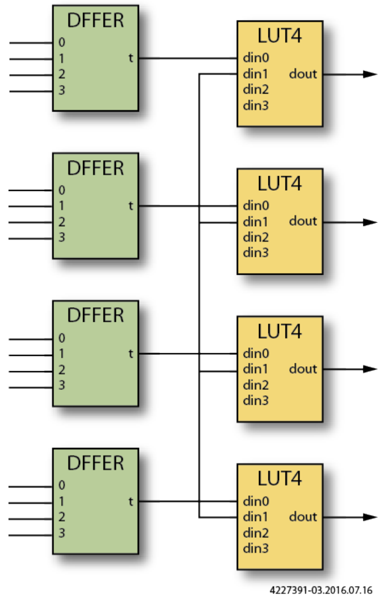


Figure 209: *Stubout Style Output Pad*

serialize_dff

The following two schematics illustrate the output of I/O virtualization using the `serialize_dff` style. Notice that the 4-bit IPIN and OPIN buses have been replaced by a single IPIN or OPIN instance. On the input side, the input pads were previously driving a bus of four DFFs. Those DFFs are now driven by the intermediate outputs of a 4-bit shift chain built from DFFs. On the output side, observe that the 4-bit output DFF shift chain is driven by four 2-to-1 MUXes. One input of the MUX comes from the flops originally driving the outputs, while the other input of each MUX is driven by the output of the previous stage of the shift chain. A new IPIN instance has been created to drive the select pin if these MUXes.

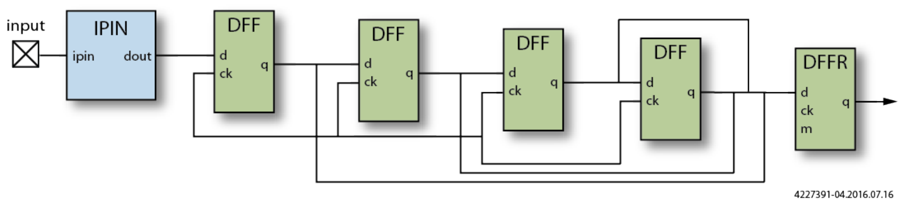


Figure 210: *serialize_dff Style Input Pad*

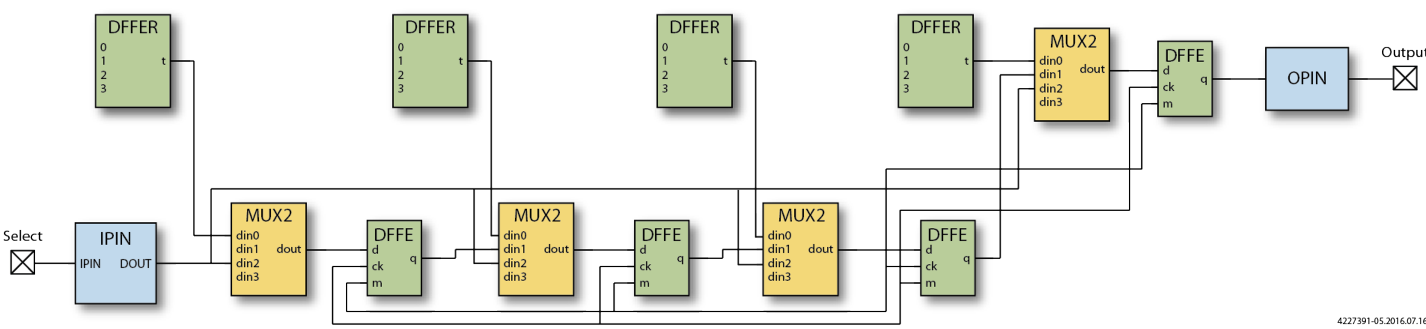


Figure 211: *serialize_dff Style Output Pad*

serialize_lut

The following two schematics illustrate the output of I/O virtualization using the `serialize_lut` style. Notice that the 4-bit IPIN and OPIN buses have been replaced by a single IPIN or OPIN instance. On the input side, the input pads were previously driving a bus of four DFFs. Those DFFs are now driven by the intermediate outputs of a 4-bit shift chain built from LUTs. On the output side, observe that the 4-bit output LUT shift chain is driven by four 2-to-1 MUXes. One input of the MUX comes from the flops originally driving the outputs, while the other input of each MUX is driven by the output of the previous stage of the shift chain. A new IPIN instance has been created to drive the select pin if these MUXes.

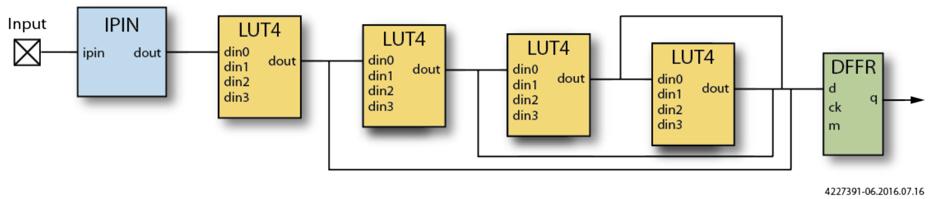


Figure 212: *serialize_lut Style Input Pad*

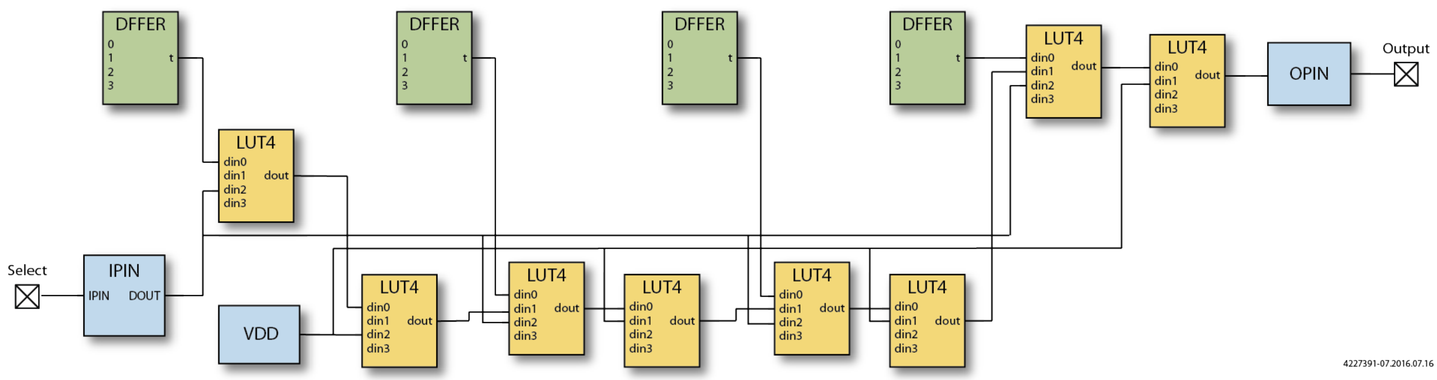


Figure 213: *serialize_lut Style Output Pad*

Managing I/Os

> **Caution!**
>
> **The I/O Assignment View is only applicable for Speedster FPGA devices**
>
> The IO Assignment View (see page 74) should be ignored when developing for other Achronix product types.

I/O electrical properties are often iteratively adjusted at the final stages of design. Frequently, it is inconvenient to alter the source RTL to make these changes, because doing so would necessitate re-running the entire flow.

The I/O Assignment View (see page 74) was created to ease these last-minute adjustments. This view allows I/O electrical changes to be made without impacting place-and-route. Electrical settings can be iteratively adjusted, the bitstream regenerated, and the design tested, repeating until the design performs as desired. The set of changed property values can then be saved off in an `.sdc` file (see Save Changed Properties Dialog (see page 168)). This `.sdc` file may be added to the project as a design constraint file, or the desired property values contained in the file could be integrated back into the source RTL.

In addition to the I/O Assignment View (see page 74), there are design rule checks that ensure all I/O instance parameters are valid. These checks are run at several points in the flow, and help to prevent creating a bitstream with any invalid I/O configurations.

Accessing Help

ACE provides a number of ways to access help information, including context-sensitive help and a built-in copy of this user guide document.

Accessing Context-Sensitive Help

ACE provides brief context-sensitive help for most parts of the application. This contextual help typically contains a brief description of the view, dialog, etc., followed by a list of hyperlinks to relevant sections within the ACE User Guide.

To cause the context-sensitive help to be shown, simply press the **F1** key in Windows, or **Shift+F1** in Linux, and the contextual help appears in a view on the right.

The following is an example of what appears when contextual help is opened while the Projects view (see page 125) has focus:

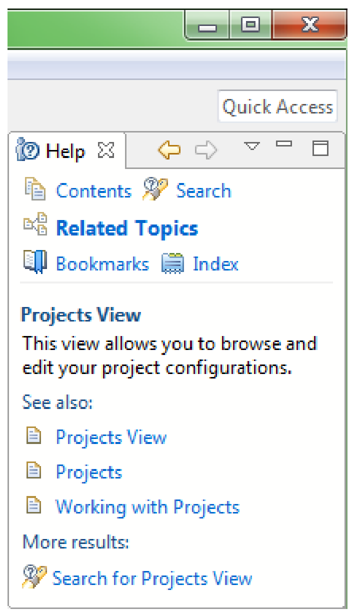


Figure 214: ***Context-Sensitive Help Example***

Navigating Help Topics

Help topics (corresponding to sections within the ACE User Guide) can be browsed using the Help window or Help view. Choosing which to use is a matter of preference; the Help view is displayed within the workbench like any other view, and is good for quick help lookups. The Help window is (as the name implies) a separate window from the rest of ACE and can be individually maximized, thus allowing easier reading of larger quantities of content.

Using the Help Window

The Help window is separate from the workbench, used exclusively for browsing and searching help content. To open the window, select **Help** → **Help Contents** from the main menu. This action opens the help window with the ( ) **Contents** tab visible in the left frame, which shows the table of contents.

> **Caution!**
>
> There is a known bug on the Linux platform in the application frameworks underlying ACE that might cause view /editor tab movements to detach instead of docking when the Help window is open. See the Troubleshooting (see page 611) section for more details, including several workarounds.

Navigating the Help Window

Table of Contents

1. In the left frame, select the ( ) **Contents** tab.
2. To find the topic to be read in the table of contents:
    a. Click to expand the subtopics.
    b. Click in the desired topic to have it displayed in the frame on the right.
3. Some topics provide links to additional related topics within (of after) their content. Click these links to learn more.
4. Use the ( ) **Back** and ( ) **Forward** buttons (above the right frame) to navigate back and forth among the recently viewed topics. These buttons behave the same way as in Web browsers.
5. Use the ( ) **Home** button (above the right frame) to return to the help home page in the ( ) **Contents**.

Searching

To quickly locate topics on a particular subject in the documentation, enter a query in the **Search** field at the top of the window. Search results are displayed in the left frame on the ( ) **Search Results** tab. For more details, see Searching Help (see page 433).

Synchronizing

Clicking the ( ) **Show in Table of Contents** button above the right frame selects that topic for that page in the **Contents** tree in the left frame (useful when navigating search results when the tree may be out of sync). The ( ) **Link with Contents** button above the left frame in the **Contents** tab keeps the navigation tree synchronized to the current topic shown in the right frame.

Maximizing and Restoring Help Frames

The two main frames of the Help window can each be maximized to take up the entire window. To maximize a frame, click the ( ) **Maximize** button in the frame toolbar, or double-click any blank part of the toolbar for that frame. To return the frame to its original size, click the ( ) **Restore** button or double-click the toolbar again.

Using the Help View

The Help view provides the same features as the Help window, but does it in a single view panel within the Workbench (see page 24) instead of in a separate window.

Searching Help

The help system includes a search engine that can run simple or complex queries on the documentation to help locate the desired information. To search help:

1. From the main menu, select **Help** → **Search.**
2. Type the word or phrase for the search subject.

3. Click **GO** or press **Enter**. The list of results are displayed below in the left frame (within the **Search Results** tab).
4. Click the topic in the list of results to view the content.

Alternately, searches can be initiated within the Help window using the **Search** field at the top left of the window.

Refining the Search Results

Reducing the Scope of the Search

Sites licensed for both Speedcore and Speedster devices can narrow the **Scope** of a search by restricting the scope to only the preferred user guide(s).

Changing the Appearance of the Search Results

Two buttons on the search results toolbar can be used to change the way results are displayed:

- The ( ) **Show result categories** button, when clicked, causes the results to be grouped by book (this action only has a noticeable effect at sites licensed for both Speedcore and Speedster devices, i.e., when both ACE User Guides are available).
- The ( ) **Show result descriptions** button, when clicked, causes a brief description of each result to be shown.

Highlighting Search Terms

By default, when a search result is selected, the search terms used to find the document are highlighted in the document content. Clicking the ( ) **Highlight Search Terms** toolbar button toggles this feature on and off. This button is available in both the help window and the help view. The state of this button is remembered in both views when displaying subsequent search results.

Search Query Syntax

Follow these expression rules for searching local help content:

- The following stop words are common English words which are ignored (not searched for) if they appear in the search expression:
  - a, and, are, as, at, be, but,
by, in, into, is, it, no, not,
of, on, or, s, such, t, that,
the, their, then, there, these,
they, to, was, will, with
- The search engine ignores character case (i.e., "Workbench" returns topics that contain "workbench", "Workbench", "WorkBench", and "WORKBENCH")
- Unless otherwise stated, there is an implied AND between all search terms so that topics that contain all the search terms are returned (e.g., "verilog module" returns topics that contain the word "verilog" and the word "module" but does not return topics that contain only one of these words)
- Use "OR" before optional terms (e.g., "project OR implementation" returns topics that contain the word "project", or the word "implementation", or both)
- Use "NOT" before terms to exclude from search results (e.g., "verilog NOT module" returns topics that contain the word "verilog" and do not contain the word "module")

> **Note**
>
> The word "NOT" only works as a binary operator (e.g., "NOT module" is an illegal search query by itself).

- Use "?" for a single-character wildcard and "*" for a multi-character wildcard (e.g., "par?" returns topics that contain "part" or "park", but not "participate", while "par*" returns topics that contain "part", "park", "participate", "pardon", etc.)

> **Note**
>
> The search engine does not accept terms with a wild card at first character position.

- Use double quotation marks around terms which should be treated as a phrase (e.g., "creating projects" returns topics that contain the entire phrase "creating projects" while topics where the words "creating" and "projects" are not consecutive are not returned)
- Punctuation acts as term delimiters (e.g., "plugin.xml" returns hits on topics that contain "plugin" and "xml", so to limit the search to the item, include the double quotes as shown)

> **Note**
>
> The search engine automatically performs "fuzzy" searches and word stemming. Entering "create" returns results including hits on topics that contain "creates", "creating", "creator", etc. To prevent the search engine from stemming a term, enclose the term in double quotes as shown.

Using the ACE SecureShare Tool to Create a Support Zip File

When encountering some problems, the Troubleshooting (see page 611) chapter and/or opening a case with Achronix Technical Support (at https://support.achronix.com/hc/en-us) might not be enough to find a solution. For these instances, ACE includes the SecureShare tool.

The Create a SecureShare Zip File dialog (see page 177) gathers all the important information from a user design and collects it into a single ZIP file. Sensitive files may be optionally excluded, or additional files included, before the ZIP file is created by the SecureShare tool. Optionally, the SecureShare tool can even encrypt the information in the ZIP file. Achronix technical support engineers can decrypt any files encrypted by the tool as they help track down and solve the problem.

To use the SecureShare tool:

1. Load (and activate) the project (see page 215) and implementation (see page 215) for which help is desired. (See also Active Project and Implementation (see page 221).) The SecureShare tool gathers the relevant files for whichever project and implementation are active when the tool is started.

> **Note**
>
> If help is needed for multiple projects or implementations, each need to be handled using separate SecureShare ZIP files.

2. Open the Create a SecureShare Zip File dialog (see page 177) by selecting **Help** → **Start SecureShare**, or by using the keyboard shortcut **Ctrl**+**Alt**+**Shift**+**S**.
3. Examine each file category and **Remove** any files containing sensitive information which should not be transmitted to Achronix.
4. **Add** any additional files which might help Achronix track down the problem. Ideally, add the files to the appropriate categories. If no appropriate category exists, add the files to the "Other" category.
5. Make sure the ZIP file (under the **Configure SecureShare** heading at the top) is pointing to an appropriate directory/filename.
6. Select the **Encrypt included files** checkbox if desired.

7. Click the **Finish** button at the bottom of the dialog.

ACE then creates a ZIP file with the chosen name in the chosen directory. If the Encrypt option was chosen, an additional file with the `.zip.encrypted` file extension is created alongside the (not-encrypted) ZIP file. The resulting `.zip` or `.zip.encrypted` file can be attached to the support request ticket.

Importing and Exporting Preferences

Preference files can be both imported to and exported from ACE, allowing individual or group preferences to be shared or migrated from an existing version of ACE to a newer version when upgrading.

Import Preferences

The Import wizard can be used to import preferences from the file system into ACE. To import a preference file:

1. Select **File** → **Import...**
2. In the Import wizard, select **Preferences** and click **Next**.

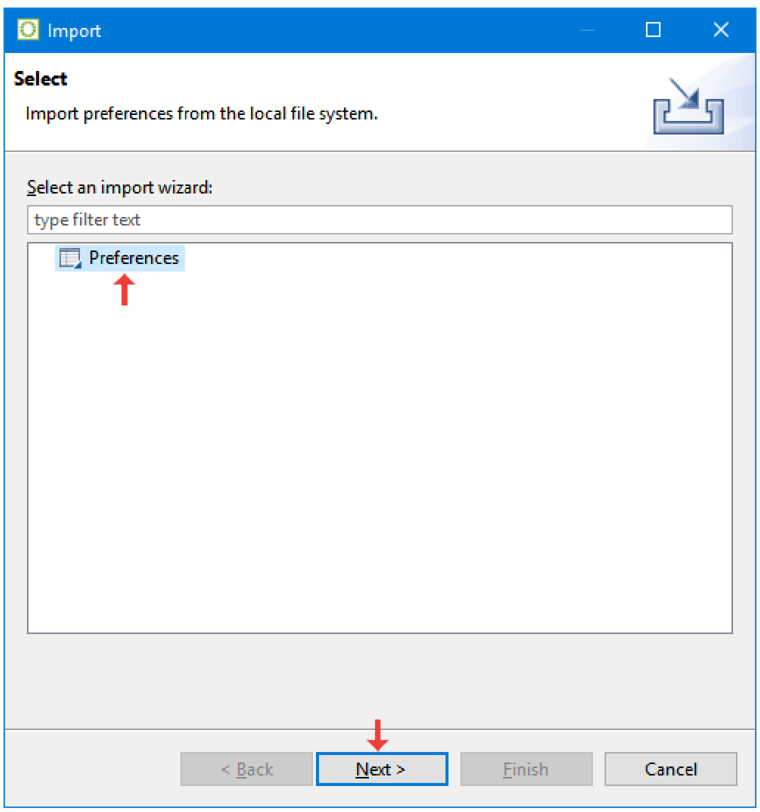


Figure 215: *Import Wizard Select Preferences Example*

3. Click **Browse...** and locate the Preferences file on the file system.
4. Select **Import all** to accept all of the preferences defined in the file.
5. Click **Finish**.

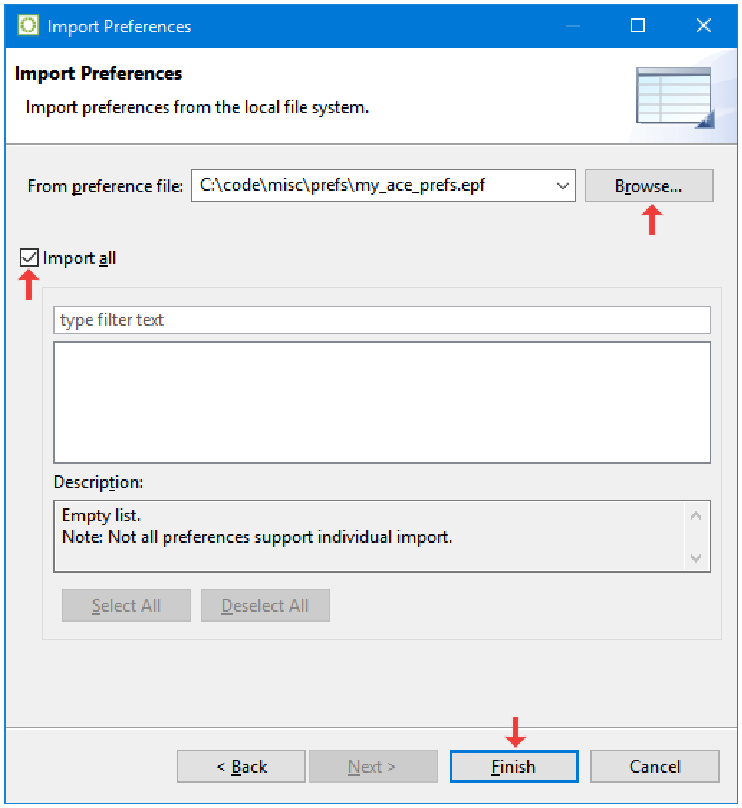


Figure 216: ***Import Wizard Locate File Example***

Export Preferences

The Export wizard can be used to export preferences from ACE to the file system. To export a preference file:

1. Select **File** → **Export...**
2. In the Export wizard select **Preferences** and click **Next**.

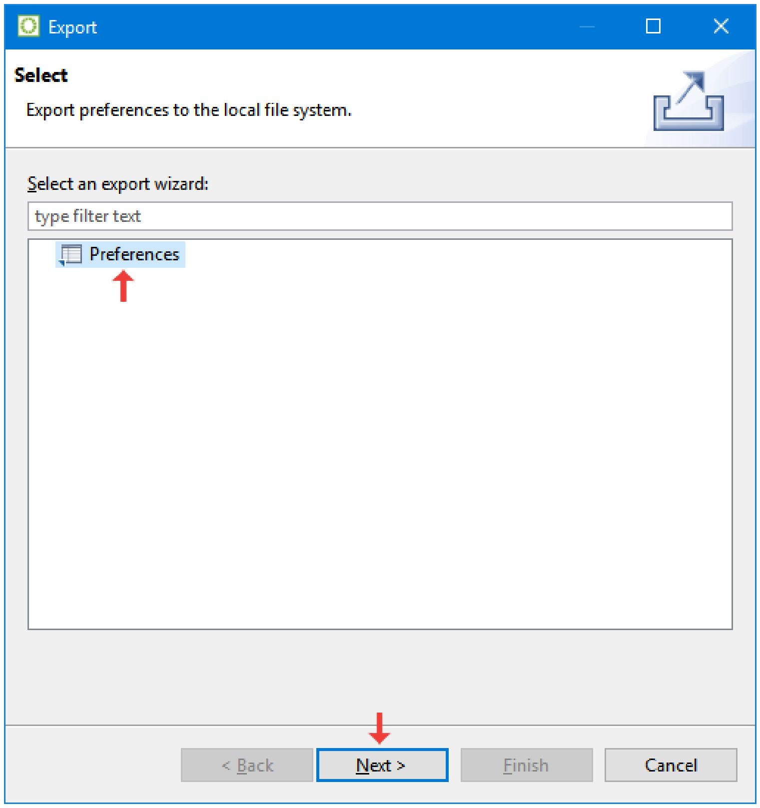


Figure 217: *Export Wizard Select Preferences Example*

3. Select **Export all** to add all of the preferences to the file.
4. Click **Browse...** and locate the preferences file on the file system.
5. Click **Finish**.

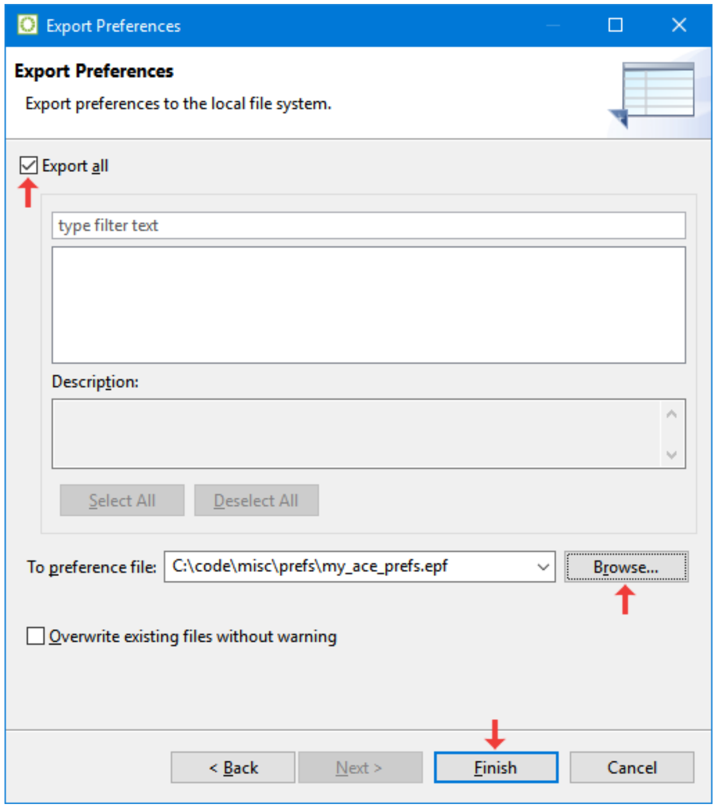


Figure 218: ***Export Wizard Locate File Example***

Plotting Serdes Rx Diagrams using JTAG

Using an open JTAG connection to the SerDes hardware, it is possible to capture Rx diagnostic data and plot Eye diagrams, Histograms, and Bathtub plots.

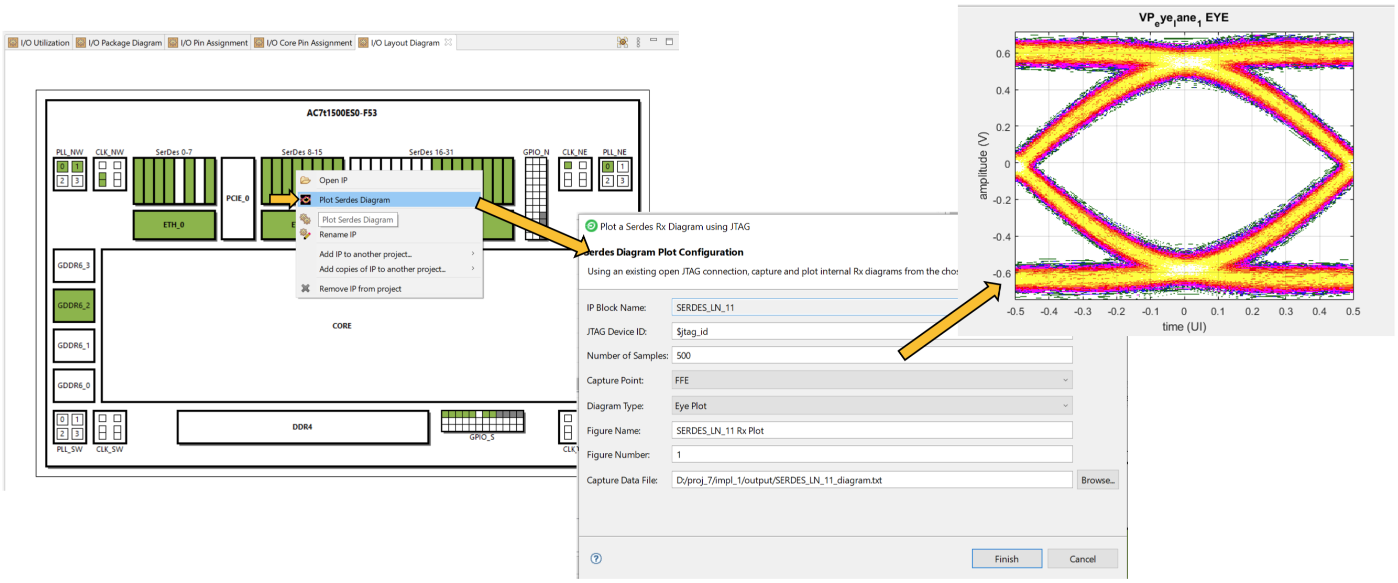


Figure 219: *SerDes Rx Eye Diagram Plot Example*

Plotting a SerDes Diagram for a SerDes Lane

Diagrams can be plotted using the following simple steps. For advanced users, the `jtag::capture_serdes_diagram_data` and `jtag::plot_serdes_diagram_data_matlab` Tcl commands may be used directly.

1. Open an ACE project containing the I/O Ring design configuration, and switch to the IP Configuration perspective.
2. Open a valid JTAG connection to the FPGA using the JTAG Tcl commands in the Tcl Console view (see page 144):

```
set jtag_id "AC12345"
jtag::open $jtag_id
jtag::ac7t1500_initialize_fcu $jtag_id
```

3. With an open JTAG connection, right-click any configured (green) SerDes Lane in the I/O Layout Diagram view (see page 72):

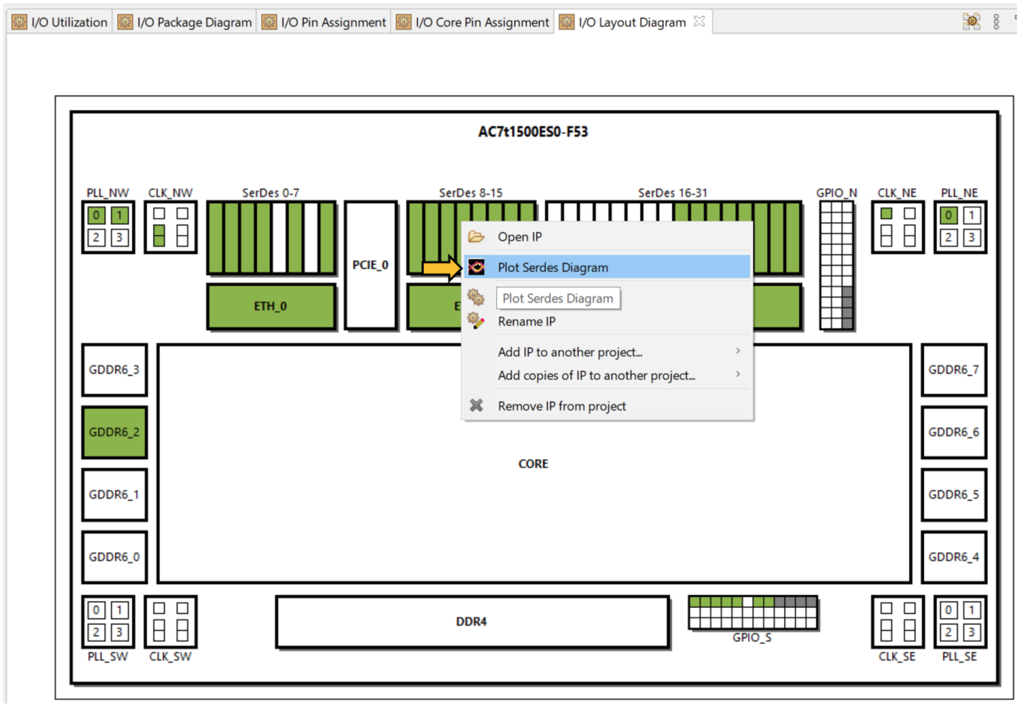


Figure 220: *Plot SerDes Diagram Lane Selection Example*

4. Configure the diagram plotting options in the Plot Serdes Diagram dialog (see page 183):

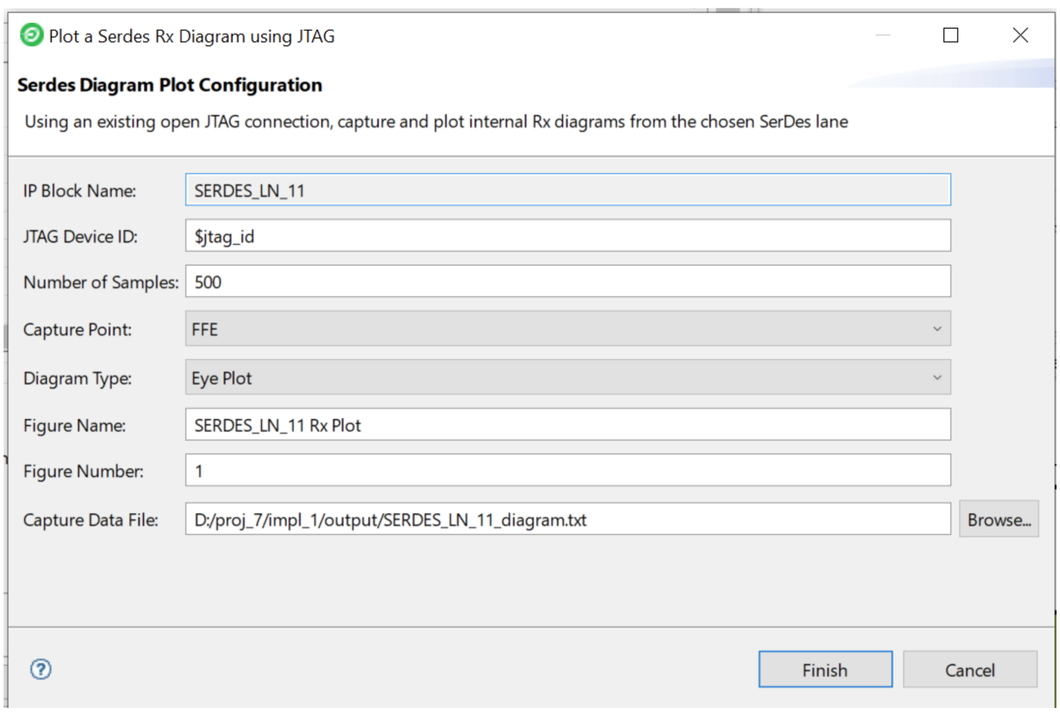


Figure 221: *Plot SerDes Diagram Dialog Configuration Example*

5. Click **Finish**.

> **Note**
>
> Multiple diagrams can be plotted from the same capture data file using the `jtag::plot_serdes_diagram_data_matlab` Tcl command after completing these steps.
>
> The diagram plotting capture over JTAG takes a long time. For 500 samples, it typically takes upwards of 10 minutes to run.

Using Partial Reconfiguration

This section begins with a high-level overview, and then continues with detailed tutorials.

- Partial Reconfiguration Tutorial (see page 442)

Partial Reconfiguration Tutorial

Partial Reconfiguration tutorial examples are provided for two different flows. ACE currently supports Partial Reconfiguration for the Speedster7t AC7t1500ES0 FPGA only.

Partial Reconfiguration Flows

- PR Flow 1: Multi-Project PR Flow using Partition Export/Import (see page 452)
- PR Flow 2: Multi-Project PR Flow using Keep Out regions (see page 477)

ACE currently supports 2 different partial reconfiguration flows. PR flow 1 is the recommended flow.

Overview of PR Flow 1

PR Flow 1 uses the ACE partitions feature. Familiarity with the ACE partition flow is recommended before proceeding with PR flow 1.

The partition import/export flow requires separate projects for the top-level design as well as each PR Core. The partition export flow must first be run in its own Synplify and ACE project for the PR core to be exported as a partition. After the partition export flow is run, ACE generates a placed and routed partition for the PR core. The partition import flow is next run in a separate Synplify and ACE project. In the partition import flow, the top-level design instantiates the previously exported PR core blackbox module and ACE imports the placed and routed data for the PR core into the top-level design.

> **Note**
>
> In the partition import flow, ACE only places and routes the top-level design and not the imported PR cores.

The base bitstream is created in the first partition import run where the previously exported PR cores are imported into the top-level design. On Silicon, the base bitstream is programmed onto the device first. The partial bitstream for the second set of PR Cores is subsequently programmed onto the device. The base bitstream includes the top-level logic as well as the PR Core logic. The partial bitstream is generated in the second partition import run where a different set of exported PR Cores are imported in the same top-level design.

Overview of PR Flow 2

The Multi-Project PR flow using keepout regions is another method to create designs that can be partially reconfigured. This flow involves using separate ACE projects to generate separate base and partial bitstreams for the top-level design and each PR core. On silicon, the base bitstream consisting of only the top-level logic is programmed first. The partial bitstreams for the other PR core can be programmed subsequently. Using this flow, the user design is stitched together at the bitstream level. There is no integration in ACE between the the top-level design and each PR core since the designs are stitched at the bitstream level. As such, this flow comes with a few pitfalls so that only high-level users are recommended to use this flow. PR flow 1 is therefore considered an improvement to PR flow 2.

Flow Comparisons

Table 152: ***Pros and Cons For Each PR Flow***

Item	PR Flow 1	PR Flow 2
Turn-around Time	**Con:** PR flow 1 has a slower turn-around time. The base bitstream consists of the top-level design as well as the PR cores. PR Cores must be exported in separate ACE sessions before being imported into the top-level design.	**Pro:** PR Flow 2 has a faster turn-around time. The base bitstream consists only of the top-level design. PR cores place-and-route can be bypassed. Only a keep-out region is needed.
Integrating Third Party Accelerators	**Pro:** PR flow 1, allows integrating third party accelerator cores at a bitstream level and also within ACE. The partition binary file for third-party accelerators can be integrated in an ACE project which allows running checks and timing analysis for the integrated design to ensure that the accelerator is compatible with the top-level design.	**Con:** PR flow 2 enables integrating third party accelerator cores only at the bitstream level. Third party accelerators cannot be tested until programmed at the bitstream level.
Bitstream Stitching	**Pro:** In PR flow 1, ACE integrates the PR core project together with the top-level project using the partitions feature. A series of checks are performed to ensure that the base and partial bitstreams can be stitched together without issue.	**Con:** In PR Flow 2, bitstreams might not be stitched together correctly if any of the ACE projects are configured incorrectly. The base bitstream and partial bitstreams are created in separate independent ACE projects so if all user projects are not set up correctly, ACE cannot catch the errors. 1. Scenario 1 – if clock nets used in the partial bitstream are not pre-routed in the base bitstream, the clock signal cannot be correctly connected to the PR cores after partial reconfiguration. 2. Scenario 2 – if clock nets on different clock tracks are pre-routed in the partial and base bitstream, the clock signals cannot be correctly connected to the PR Cores after partial reconfiguration. 3. Scenario 3 – if keep-out regions are not set on clusters in the base bitstream that are used by PR cores, the top-level logic might be lost after partial reconfiguration.
Closing Timing	**Pro:** In PR flow 1, ACE integrates the PR core project together with the top-level project during the partition import run and also performs timing analysis for the integrated design.	**Con:** In PR Flow 2, the design might not meet timing on paths between the top-level and PR core after the partial bitstream is programmed because timing analysis is performed separately in projects for the top-level design and PR cores (timing delays are different on the east and west sides of the Speedster7t AC7t1500ES0 FPGA.)
Simulation	**Pro:** Simulating the entire design is possible in the partition import run.	**Con:** Simulating the entire design within an ACE project is difficult because there is no integration between separate ACE projects. Though not impossible, a lot of manual work is required.

Flow Diagrams

The following sequence outlines the high-level steps needed to partially reconfigure PR core A as PR core B.

PR Flow 1: Multi-project PR Flow Using Partition Export/Import

In the operating procedure for this example, the base bitstream in step 3 is programmed onto silicon first. This base bitstream includes the top-level logic and the PR Core A logic. To partially reconfigure PR Core A with PR Core B, the partial bitstream created in step 6 is programmed subsequently.

1. Export PR core A as a partition using ACE.

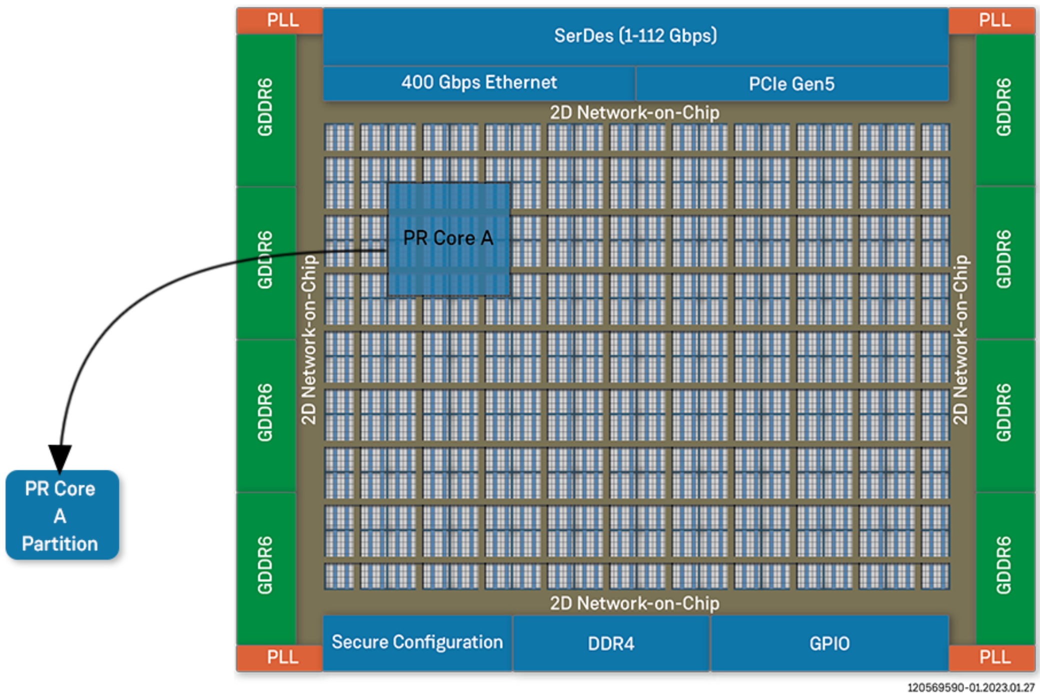


Figure 222: ***Export PR Core as a Partition Example***

2. Import PR core A partition into top-level design.
3. Generate a base bitstream for the entire device.

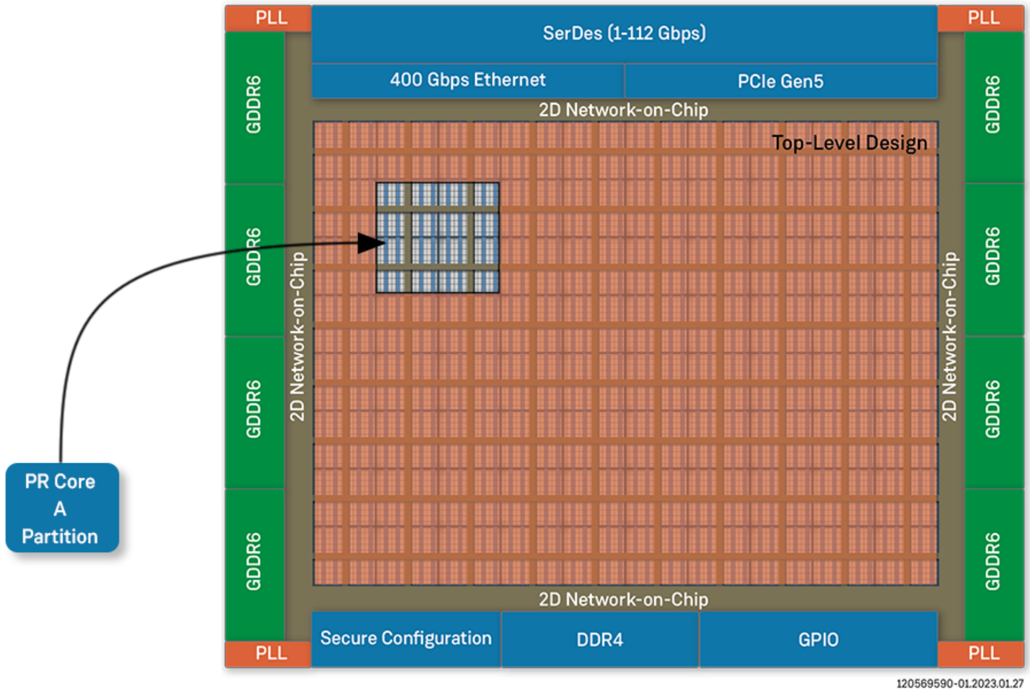


Figure 223: ***Import PR Core to Top Level Example***

4. Export PR core B as a partition using ACE.

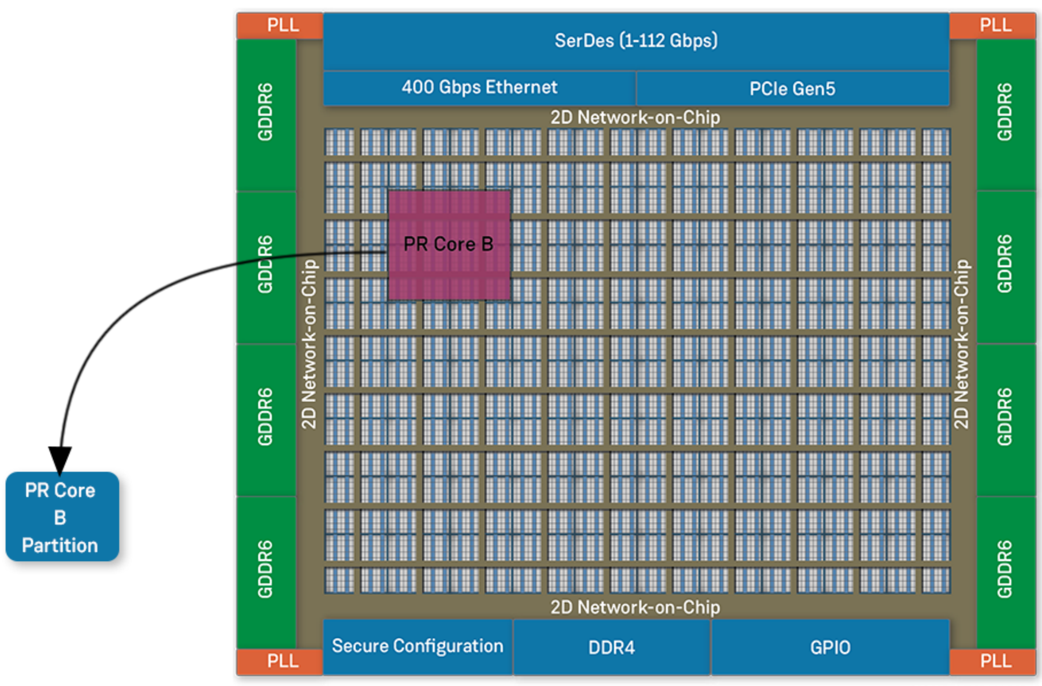


Figure 224: ***Export PR Core as a Partition Example***

5. Import the PR core B partition into the same top-level design.
6. Generate a partial bitstream only for PR core B.

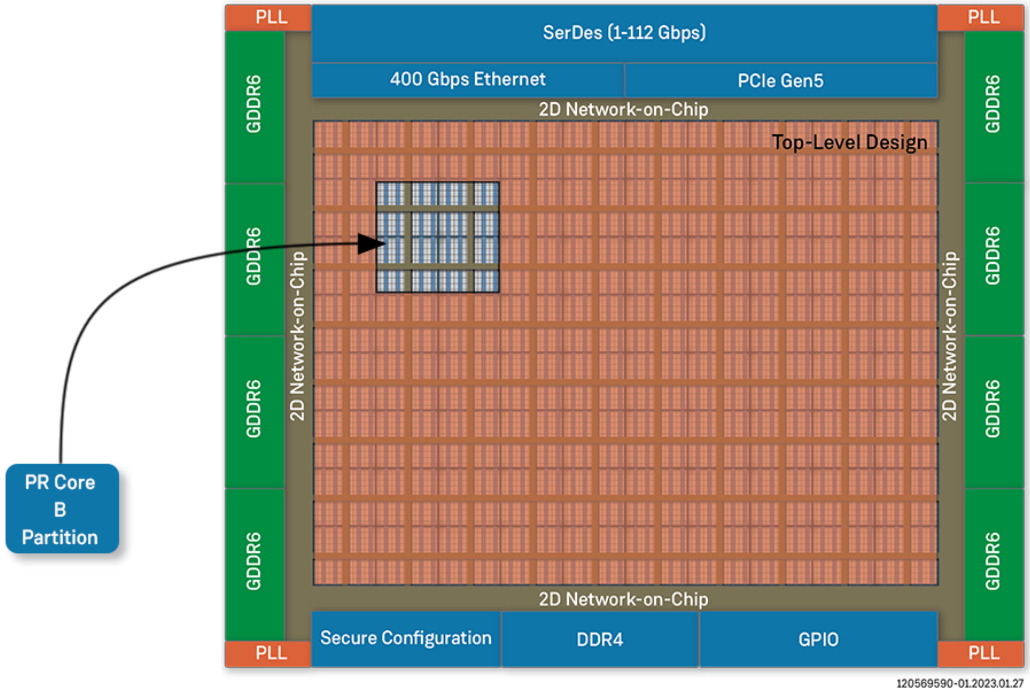


Figure 225: *Generate Partial Bitstream Example*

PR Flow 2: Multi-project PR Flow Using Keep Out Regions

In the operating procedure for this example, the base bitstream created in step 3 is programmed onto silicon first. Unlike PR Flow 1, the base bitstream only contains the top-level logic (not the PR Core A logic). The partial bitstream for PR Core A (created in step 1) is programmed next. To partially reconfigure PR Core A with PR Core B, the partial bitstream for PR Core B (created in step 2) is programmed subsequently.

1. Generate a partial bitstream for PR core A using ACE.

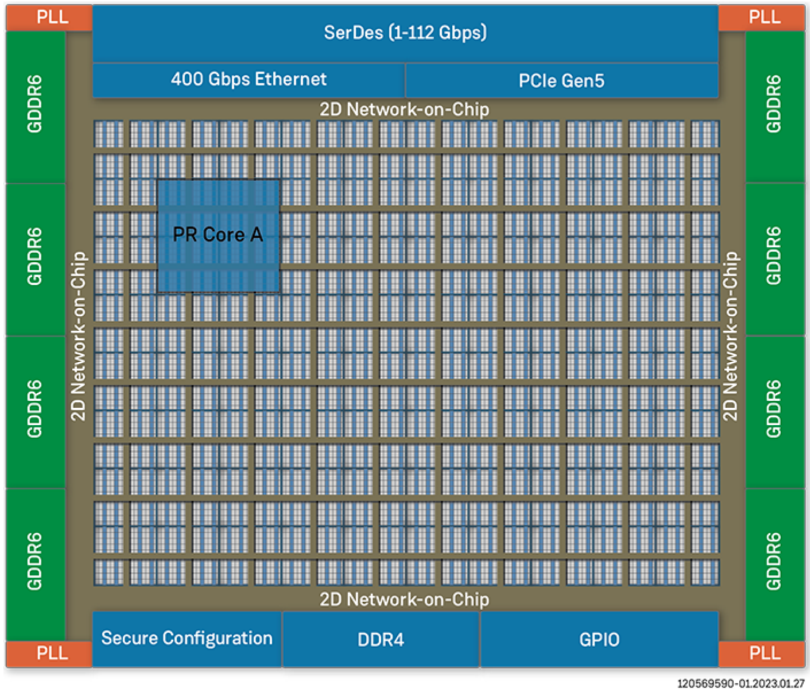


Figure 226: *Generate Partial Bitstream Example*

2. Generate a partial bitstream for PR core B using ACE.

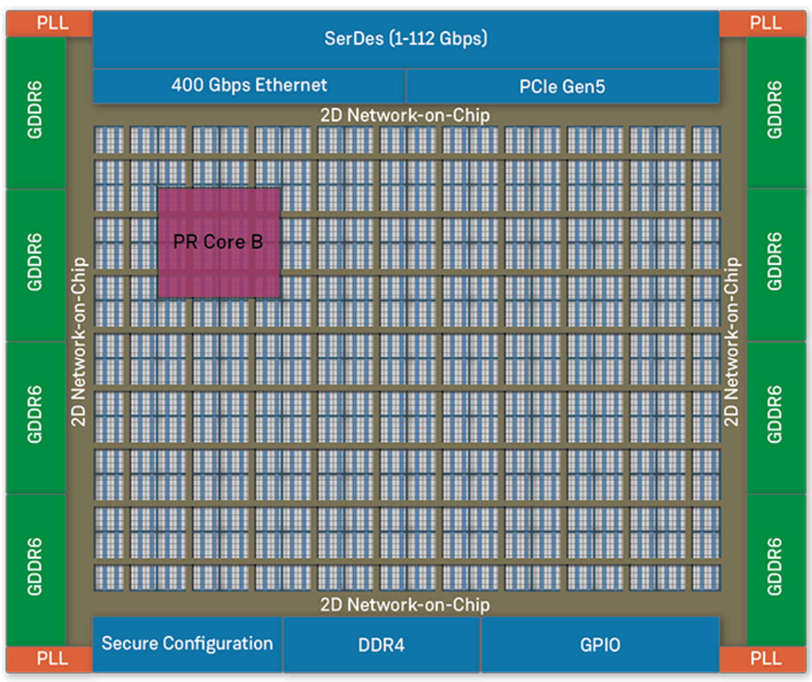


Figure 227: *Generate Partial Bitstream Example*

3. Generate a base bitstream for the top-level design using ACE with keep out regions for the PR cores.

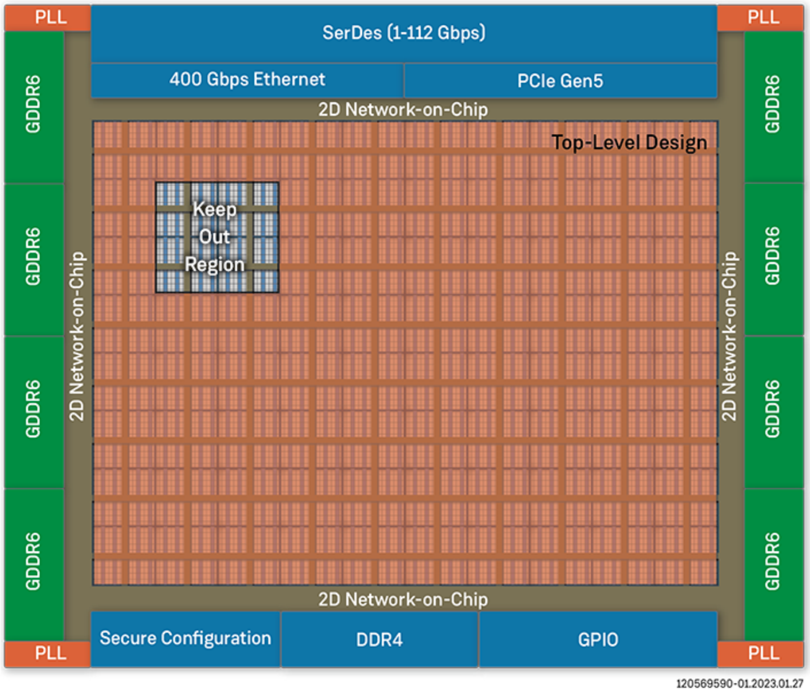


Figure 228: *Generate Bitstream Example*

Design Details

This tutorial uses a simple example design to illustrate running a design through PR Flow 1. In this tutorial, three separate bitstreams are created:

1. Base Bitstream (Top-level design)
2. Partial Bitstream 1 (PR Core 1)
3. Partial Bitstream 2 (PR Core 2)

Base Bitstream (Top-level design)

RTL files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/rtl/...`
2. PR flow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src/rtl/...`

Constraint files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/constraints/...`
2. PR flow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src/constraints/...`

The top-level design instantiates the user static logic. Static Logic refers to any user design logic which is not intended to be reconfigured in the PR flow. The base bitstream is generated using the top-level design. The top-level design in this tutorial demonstrates a 2-bit binary up-counting LED display to indicate that the board and FPGA are operating properly upon powerup. The RTL along with the device-specific netlists and constraints are available under the <ACE_install_dir>/Achronix/examples/partial_reconfiguration directory.

Other than just instantiating the user static logic, the top-level design also must instantiate any shared system controller logic, I/O Ring, and I/O interfaces used in the other PR cores. Clock nets used in each PR core to be partially reconfigured must also be carefully pre-routed since clock nets are also routed on the core fabric HW clock network, which is not reconfigurable. Also, a keep-out region for each PR Core must be created to prevent any logic and routing from the top-level design from entering the cluster(s) used by the PR Cores in the partial bitstreams.

Partial Bitstream 1 (PR Core 1A)

RTL files:

1. PR flow 1 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/rtl/...
2. PR flow 2 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/rtl/...

Constraint files:

1. PR flow 1 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/constraints/...
2. PR flow 2 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/constraints/..

PR core 1 consists of a 32-bit adder with outputs connected to a 32-bit register. This 32-bit register is connected to the NAP initiator in the cluster. The NAP Initiator can be used to read the 32-bit adder output register via the 2D NoC with the JTAG interface or with PCIe. In this tutorial, the JTAG interface is used to read the user registers. There is also a reset register that can be written to in this design. The reset register drives all resets in PR core 1.

Partial bitstreams must not contain any I/O ring programming. Clock nets also must be pre-routed on the clock network on a specified track number. The top-level design must also pre-route the same clock net on the same track number to the same cluster where PR core 1 is placed. A cluster map also must be set to identify which core fabric cluster(s) are to be re-programmed. The following sections in this tutorial cover this in more detail.

Partial Bitstream 1 (PR Core 1B)

RTL files:

1. PR flow 1 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b/src/rtl/...
2. PR flow 2 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1b/src/rtl/...

Constraint files:

1. PR flow 1 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/constraints/...
2. PR flow 2 – <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/constraints/..

PR core 1B is identical to PR core 1A except that the 32-bit adder increments by 2 instead of 1 during each read.

Partial Bitstream 2 (PR Core 2A)

RTL files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/src/rtl/...`
2. PR flow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/src/rtl/...`

Constraint files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/src/constraints/...`
2. PR fFlow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/src/constraints/...`

PR core 2A is identical to PR core 1A except that the 32-bit adder instead is implemented as a 32-bit subtractor. The 32-bit subtractor decrements by 1 at each NAP read.

Partial Bitstream 2 (PR Core 2B)

RTL files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/src/rtl/...`
2. PR flow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b/src/rtl/...`

Constraint files:

1. PR flow 1 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/src/constraints/...`
2. PR flow 2 – `<ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b/src/constraints/...`

PR core 2B is identical to PR core 2A except that the 32-bit subtractor now decrements by 2 instead of 1 at each NAP read.

Reset Configuration Options

This tutorial uses option 1, below, but there are several ways resets for a partial reconfiguration design can be configured. The different methods are listed below, ordered by recommendation (1 being the most and 4 being the least recommended method).

1. Use a NAP to control reset via the 2D NoC. This can be performed using host software (e.g., `jtag::nap_axi_write` or PCIe), or top-level control logic.
   This reset scheme is utilized by the demo in this tutorial. All nets driven by reset are self contained within the PR zone.
2. Use a DFF chain to de-assert reset when the cluster enters user mode.
   - Use init values on DFFs to toggle the reset when a cluster enters user mode.

- This is an example showing how to instantiate a DFF chain for an active-low reset design:

```
wire rst_wire_1;
wire rst_wire_2;
wire i_rst;

// First DFF in chain
(* must_keep=1 *) ACX_DFF #(
    .init   (1'b1)
) reset_dff_1 (
    .q      (rst_wire_1),
    .d      (),
    .ck     (i_clk)
);


// Second DFF in chain
(* must_keep=1 *) ACX_DFF #(
    .init   (1'b0)
) reset_dff_2 (
    .q      (rst_wire_2),
    .d      (rst_wire_1),
    .ck     (i_clk)
);


// Third DFF in chain (q-pin drives the PR Core reset)
(* must_keep=1 *) ACX_DFF #(
    .init   (1'b0)
) reset_dff_3 (
    .q      (i_rst),
    .d      (rst_wire_2),
    .ck     (i_clk)
);
```

- In this example, `reset_dff_3` drives the reset with `1'b0` for 2 clock cycles before being de-asserted to `1'b1`.
- The disadvantage of this reset scheme is that the resets cannot be toggled by the user.

3. Use top-level CLK I/O pads to drive the reset to the global clock trunk using clock-preroute constraints.
   - Pre-route the reset net on a specified track on the global clock trunk.
   - PDC constraints:

```
#Create a clock IPIN for your reset and place it on a clock IPIN tile
create_boundary_pins {p:i_reset_n} {i_reset_n_ipin} -clock
set_placement -fixed -batch {p:i_reset_n} {d:i_user_06_00_trunk_00[16]}


#Pre-route the reset net on clock track 1 to the PR Zone defined for your PR Core.
#add_clock_preroute "i_reset_n_ipin_net" 1 -placement_regions "PR_ZONE_2_7"
```

   - Disadvantage: The reset net is routed over the clock network so there is one less clock track available.
4. Use top-level Data I/O pads to drive the reset and use the `data2clk` path to route the reset to the global clock trunk using clock-preroute constraints.

- Pre-route the reset net on a specified track on the global clock trunk.
- PDC constraints

```
#Create a data IPIN for your reset and place it on a data IPIN tile
create_boundary_pins {p:i_reset_n} {i_reset_n_ipin}
set_placement -fixed -batch {p:i_reset_n} {d:i_user_11_09_lut_13[15]}


#Pre-route the reset net on clock track 1 to the PR Zone defined for your PR Core.
#add_clock_preroute "i_reset_n_ipin_net" 1 -placement_regions "PR_ZONE_2_7"
```

- Disadvantages: There must be a path available from the data IPIN to the central clock trunk. If there are no paths available because all clusters are blocked off and in use, the router cannot pre-route this reset net.

PR Flow 1: Multi-Project PR Flow using Partition Export/Import

Introduction

The Multi-Project flow with partition export/import is the recommended flow for partial reconfiguration.

The partition import/export flow requires separate projects for the top-level design as well as for each PR core. The partition export flow must first be run in its own Synplify and ACE project in order for a PR core to be exported as a partition. After the partition export flow is run, ACE generates a placed-and-routed partition for the PR core. The partition import flow is next run in a separate Synplify and ACE project in which the top-level design instantiates the PR core blackbox module that was previously exported. ACE imports the placed-and-routed data for the PR core into the top-level design.

> **Note**
>
> In the partition import flow, ACE does not place-and-route the imported PR cores, only the top-level design.

Moving Partitions

Partitions are also very useful for replicating and moving logic. Each PR core can be placed-and-routed, and the timing-closed, once in one location of the fabric, and then stamped down multiple times and relocated throughout the fabric. This is accomplished by instantiating multiple copies of the exported PR core partition in the top-level design. ACE then imports the timing-closed PR core partition database. Initially, each instance of the PR core is placed on top of each other but ACE has a special feature which allows partitions to be moved/re-mapped to other locations in the fabric. This is made possible using relative placement and routing to map to the repetitive cluster-based architecture of the fabric.

High-Level Design Flow

The bottom-up flow must be used for PR flow 1. This involves exporting a partition for each of the PR cores and importing them into the top-level design. The base bitstream can be generated for the top-level design to contain each of the PR cores. Subsequent partial bitstreams can be generated from the top-level design project or from each PR core project.

Please follow this high-level design flow carefully:

1. Export Partition for PR Core 1A (see page 453)
    a. Synthesize PR Core 1A Using Synplify (see page 453)
    b. Run PR Core 1A Through Run Prepare in ACE (see page 455)
    c. Set Up Region Constraints for PR Core 1A (see page 456)
    d. Set Up Clock Pre-Routing Constraints for PR Core 1A (see page 457)

Export Partition for PR Core 1A

Synthesize PR Core 1A Using Synplify

1. Navigate to the following directory:
   `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src`

   > **Note**
   >
   > `<ace_install_dir>` is the directory path where ACE was installed.

   Below this directory should be the following two directories:

   - `rtl`

- `constraints`

2. In the `rtl` directory, create a new Synplify project and add all of the RTL files.
3. Add the `pr_core_1a_timing.sdc` file to the `constraints` directory.
4. Create a compile point for the PR core 1 module to be exported as a partition.

> **Note**
>
> The compile point cannot be set for the top-level module.

5. Add the `pr_core_1a_export_partitions.fdc` file from the `constraints` directory to the project. The compile point is set in the file as follows:

```
define_compile_point {v:work.pr_core_1a} -type {locked}
```

6. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
7. Set **Part** to **AC7t1500ES0**.
8. Set **Package** to **F53**.
9. Set **Speed** to **C2**.
10. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_core_1a_top**.
11. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_core_1a_top**.
12. Set **Include Path Order** to **<ace_install_dir>/libraries**.
13. Add the AC7t1500ES0_synplify.sv file to the project, this file is under <ace_install_dir>/libraries/device_models.

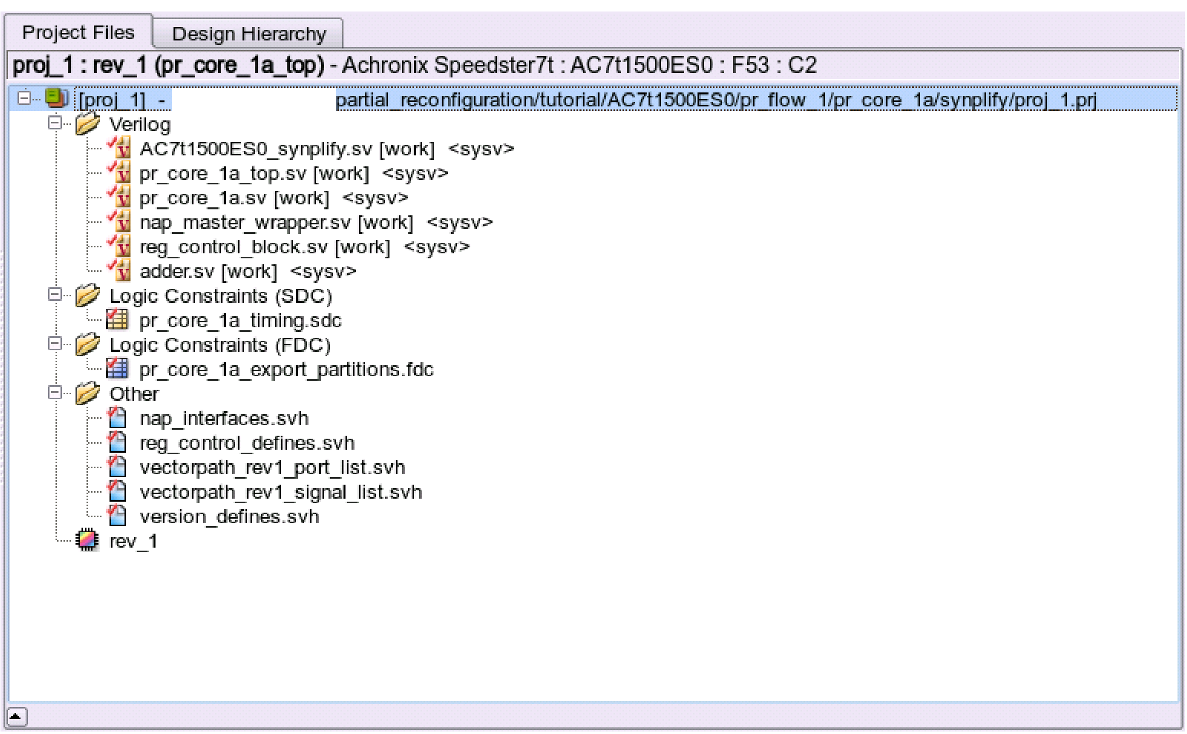


Figure 229: ***Project Example***

14. Run and compile the design using Synplify.
15. After successful completion, Synplify generates a gate-level netlist under `<synplify_out_dir>/rev_1/pr_core_1a_top.vm`.
16. Synplify also creates a `.prt` file indicating that the compile point was successfully set for PR core 1A. This file is generated under `<synplify_out_dir>/rev_1/pr_core_1a_top_partition.prt`.

Run PR Core 1A Through Run Prepare in ACE

1. In the **Projects** view, click the ( ) **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the **Projects** view, click the ( ) **Add Source Files to Project** button.
4. Add the following files to the project:
   - `pr_core_1a_top_partition.prt` (the partition info file created by Synplify in step 1a)
   - `pr_core_1a_top.vm` (the synthesized netlist created by Synplify in step 1a)
   - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/ioring_design/pr_core_1a_ioring.pdc`
   - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/ioring_design/pr_core_1a_ioring.sdc`
5. In the **Options** view, navigate to the **Design Preparation** section and ensure that the **Export All Partitions** option is checked and all other options are correctly set.
6. Copy the option values as shown in the following figure.
7. In the **Flow** view, right-click **Run Prepare** and select **Run Selected Flow Step**:

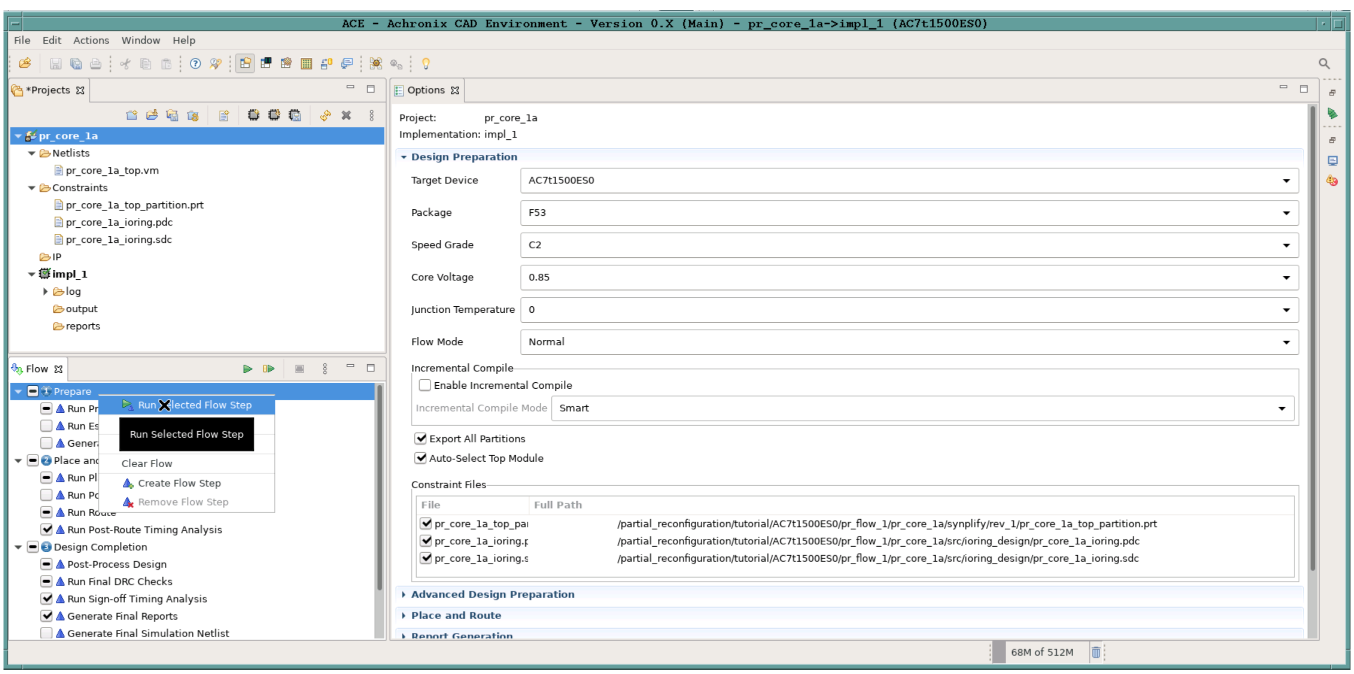


Figure 230: *Run Prepare Flow Step Example*

Set Up Region Constraints for PR Core 1A

1. Ensure that the run_prepare flow step has completed successfully.
2. On the **Floorplanner** view, click the **Placement Region Tool** button (highlighted in red in the following image).
3. Click and drag within the Floorplanner to create a PR zone for the PR core.

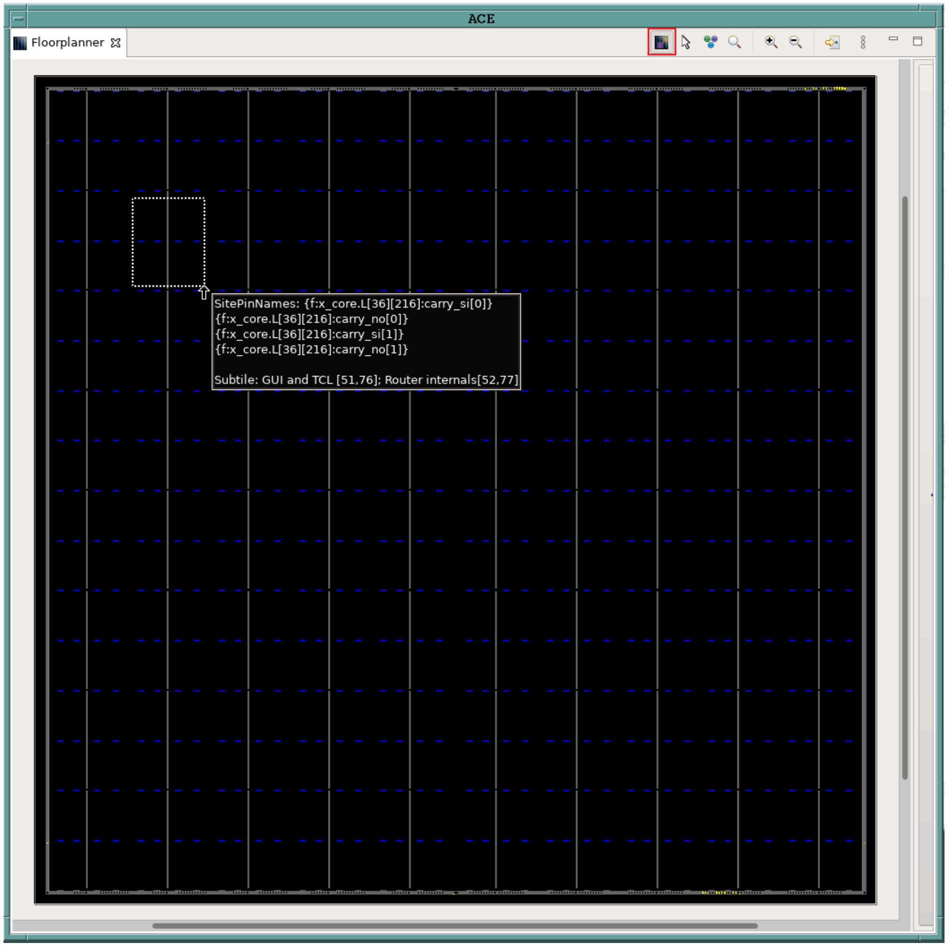


Figure 231: ***Creating a Placement Region Zone Example***

4. After releasing the left mouse button, the **Create Placement Region** dialog appears.
5. In **Region Name**, enter a name for the PR zone.
6. Set **Region Alignment** to **Snap to Fabric Clusters** (a "fabric cluster" represents the lowest level of granularity at which the core fabric can be partially reconfigured).
7. Set the **Region Type** to **Inclusive** (ensures that all instances are constrained to the region).
8. Select **Include Routing** (ensures that all routed nets are constrained to the region).
9. Select **Is Partial Reconfiguration Zone** (indicates this region is a PR zone).

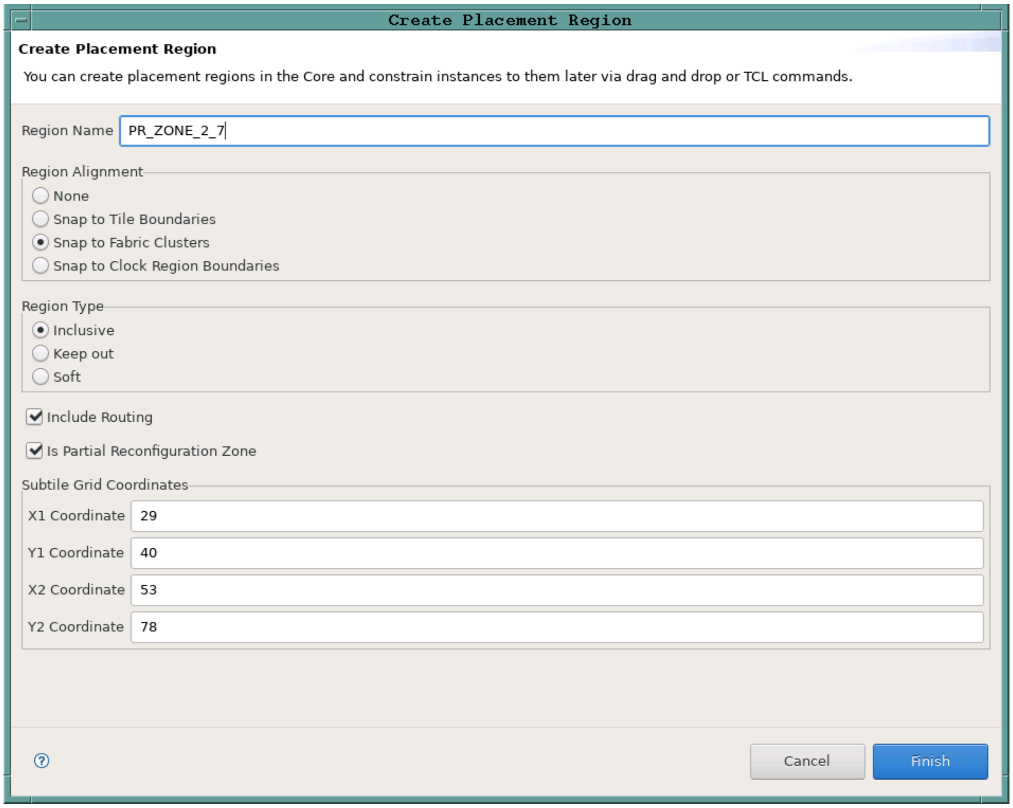


Figure 232: *Create Placement Region Dialog Example*

10. Click **Finish**. The Tcl console displays the following:

```
create_region "PR_ZONE_2_7" {29 40 53 78} -snap fabric_clusters -type inclusive -include_routing -
pr_zone
```

11. Add all instances in the PR core to the region. A recommended quick way of doing so is to use the `add_region_find_insts` Tcl command nested with the `find` Tcl command. Run the following example in the Tcl Console to add all instances in PR core 1A to the PR zone (`i_pr_core_1a` is the instance name for the top-level partial reconfigurable module).

```
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts}
```

Set Up Clock Pre-Routing Constraints for PR Core 1A

1. In the **Partitions** view, right-click the cell below the **Clock Pre-Routes** column heading and select **Configure Clock Pre-Routes** as shown:

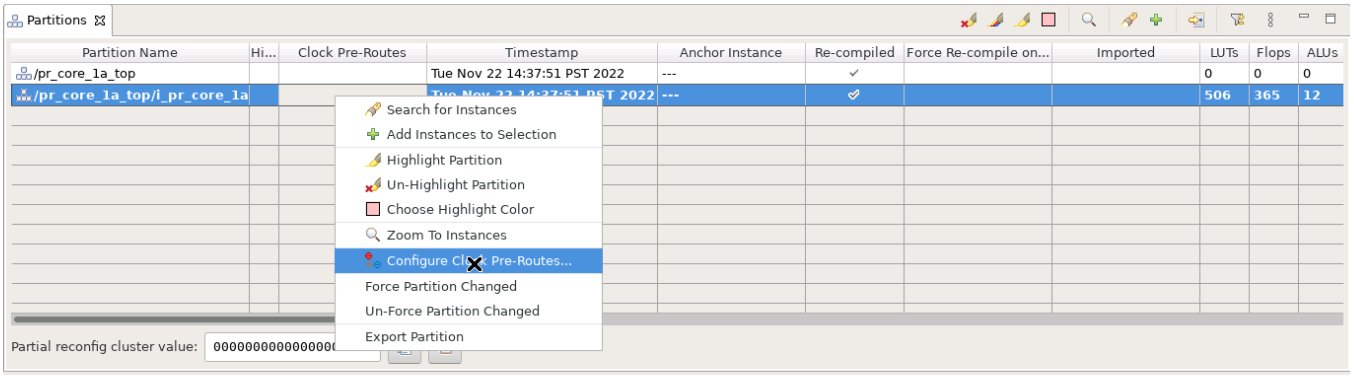


Figure 233: *Configure Clock Pre-Routes Example*

2. The **Configure Clock Pre-Routes** dialog appears with the `i_clk_pr_1_ipin_net` signal that drives the PR core 1A clock pins pre-routed on clock track 1.

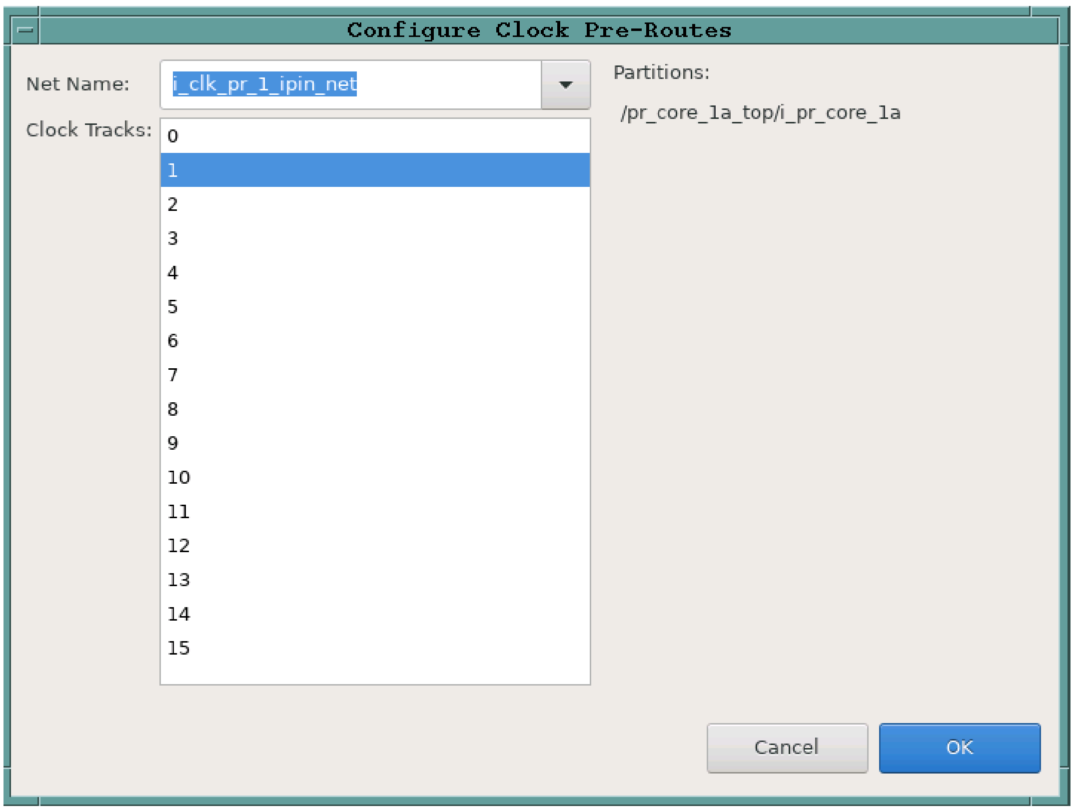


Figure 234: *Configure Clock Pre-Routes Dialog Example*

3. Click **OK**. The Tcl console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -partitions { /pr_core_1a_top/i_pr_core_1a }
```

Run PR Core 1A Through Place-and-Route in ACE

1. In the **Flow** view, right-click **Place and Route** and select **Run Selected Flow Step** as shown:

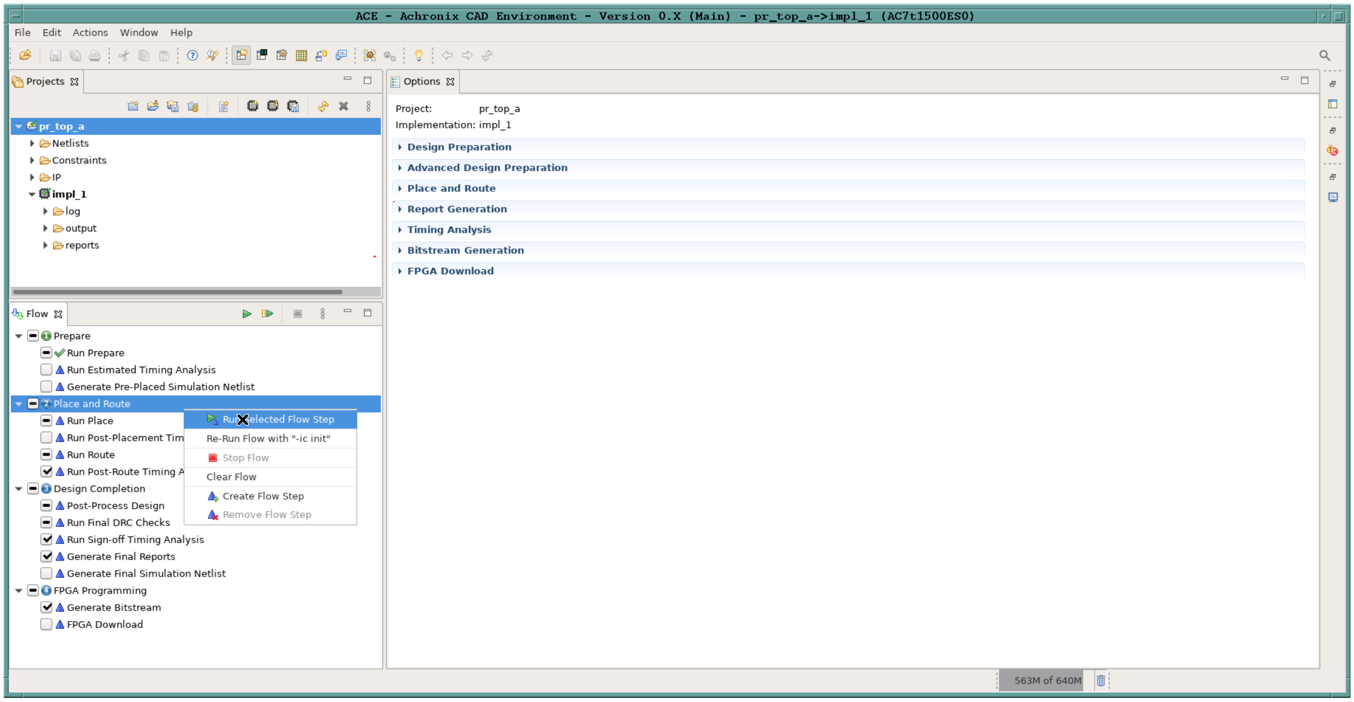


Figure 235: ***Run Place and Route Flow Step Example***

Run Final Sign-Off Timing Analysis for PR Core 1A

1. Ensure that the place-and-route flow step completed successfully.
2. In the **Flow** view, right-click **Design Completion** and select **Run Selected Flow Step** as shown:

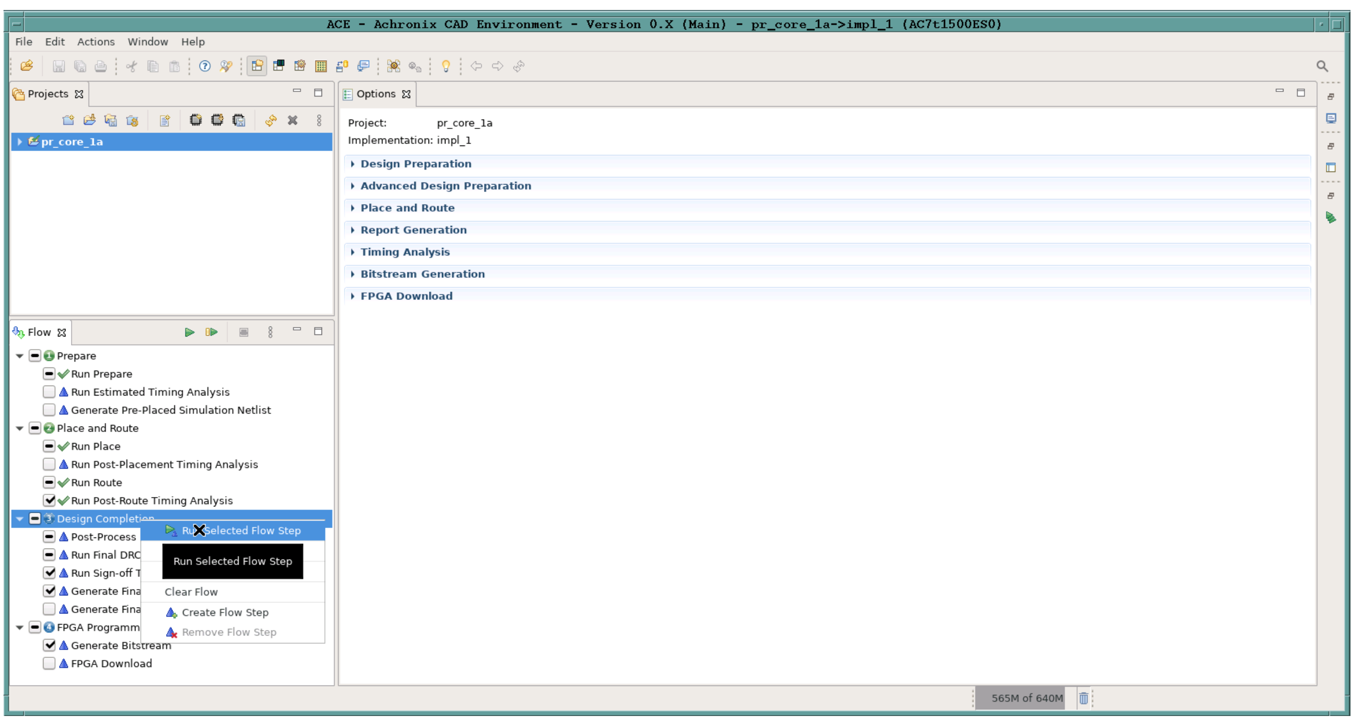


Figure 236: ***Run Design Completion Flow Step Example***

Save Region/Clock Pre-Routing Constraints Back to PDC File

1. Ensure that the Design Completion flow step completed successfully.
2. In the **Placement Regions** view, click the ( ) **Save Placement Regions** button.

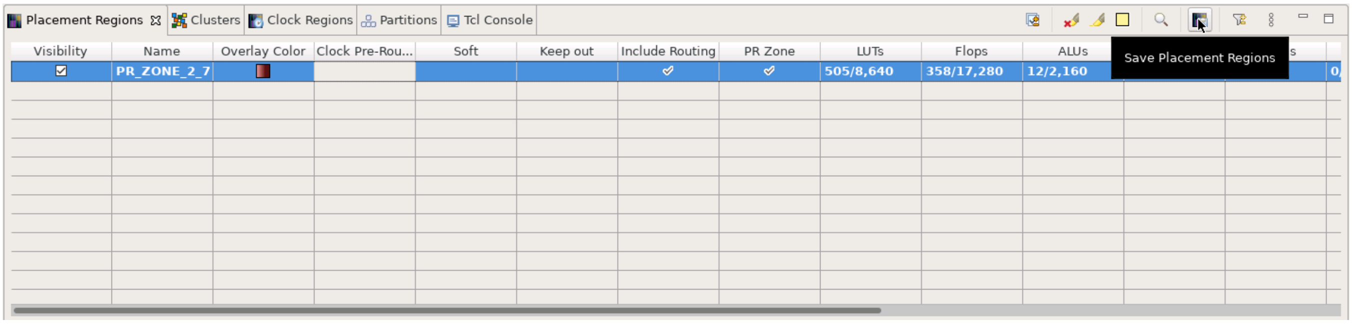


Figure 237: ***Save Placement Regions Example***

3. The **Save Placement Regions** dialog appears.
4. Click **Finish** to save the PR zone region constraints.
5. The Tcl console displays the following and the `placement_regions.pdc` file is created.

```
create_region "PR_ZONE_2_7" {32 41 87 92} -snap fabric_clusters -type inclusive -include_routing -
pr_zone
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

6. ACE automatically generates a clock pre-route file to the output area under `<ace_output_dir>/ace/impl_1/output/partial_reconfig_core_1_top_clock_preroutes.pdc`.

> **Note**
>
> This file should not be added to the ACE project as it is regenerated during each run. Make a copy of this file before adding it to the project.
>
> Alternately, use the `save_clock_preroute` Tcl command to generate a `.pdc` file with the clock pre-route constraints.

Ensure ACE Generated Blackbox File and Exported Partition for PR Core 1A

ACE generates a blackbox file for the exported partition (PR core 1A). Ensure that this file exists in the following directory:

```
<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output
/blackboxes/pr_core_1a_bb.v
```

ACE also exports a partition database file for the exported partition (PR core 1a). Ensure this file exists in the following directory:

```
<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output
/partitions/pr_core_1a_top.i_pr_core_1a.epdb
```

Export Partition for PR Core 2A

Repeat Procedure for PR Core 2A

1. Repeat the steps in section 1 to export a partition for PR core 2A.
   PR core 2A is placed on a different cluster (row=7, col=6) than PR core 1A. Steps 1c, 1d and 1g are skipped by directly adding the following file to the project:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a
/pr_core_2a_ace_placements.pdc
```

   This file contains all the correct region and clock pre-route constraints.

Importing PR Core 1A and 2A Partitions and Generating Base Bitstream for Top-Level Design

Create I/O Ring Configuration for Target Board

In this design, a simple clock input is driving a PLL to create clocks for the core fabric. Also, a simple GPIO interface to display LEDs is in the top-level design.

1. Navigate to the **IP Libraries** view.
2. Right-click the **Clock I/O Bank** IP and select **New IP Configuration**:

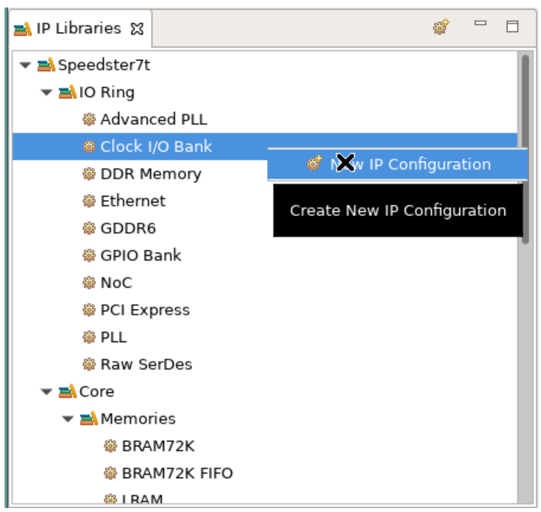


Figure 238: ***New IP Configuration Example***

3. Configure the Clock I/O Bank for the PR cores using the options shown in the following image:

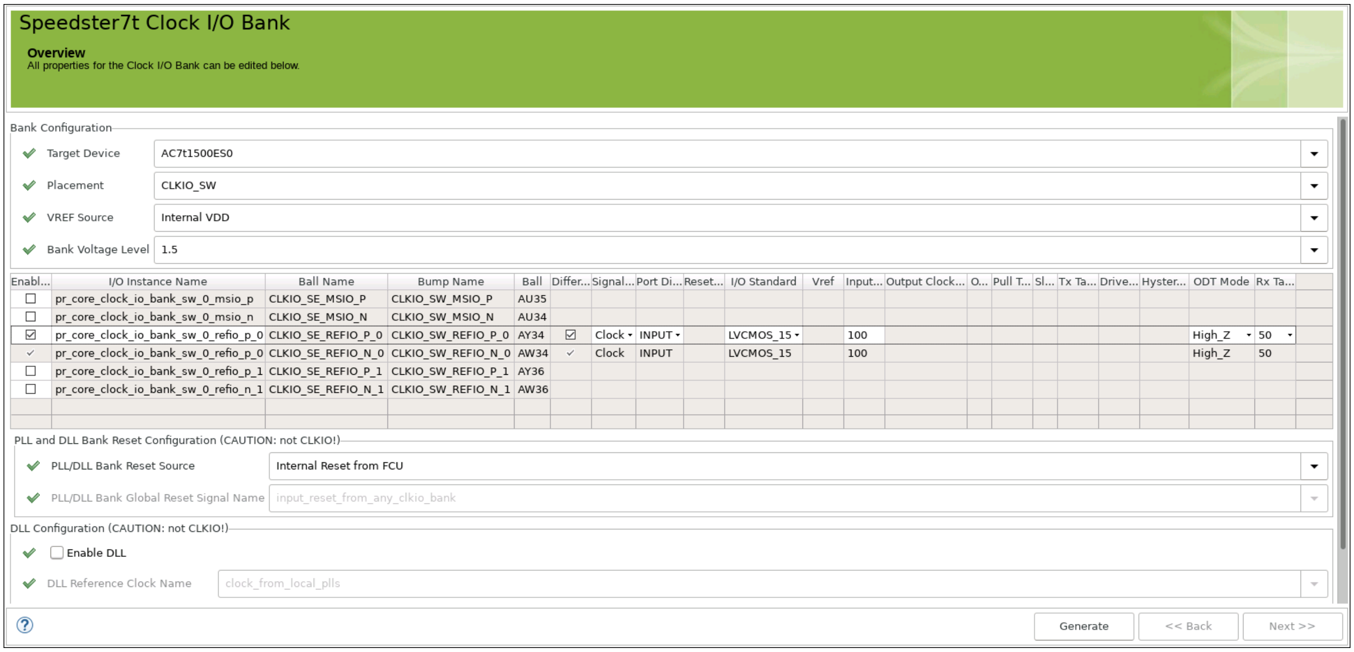


Figure 239: ***Clock I/O Bank Configuration Example***

4. In the **IP Libraries** view, right-click the **PLL** IP and select **New IP Configuration**.
5. Configure the PLL for the PR core using the options shown in the following image:

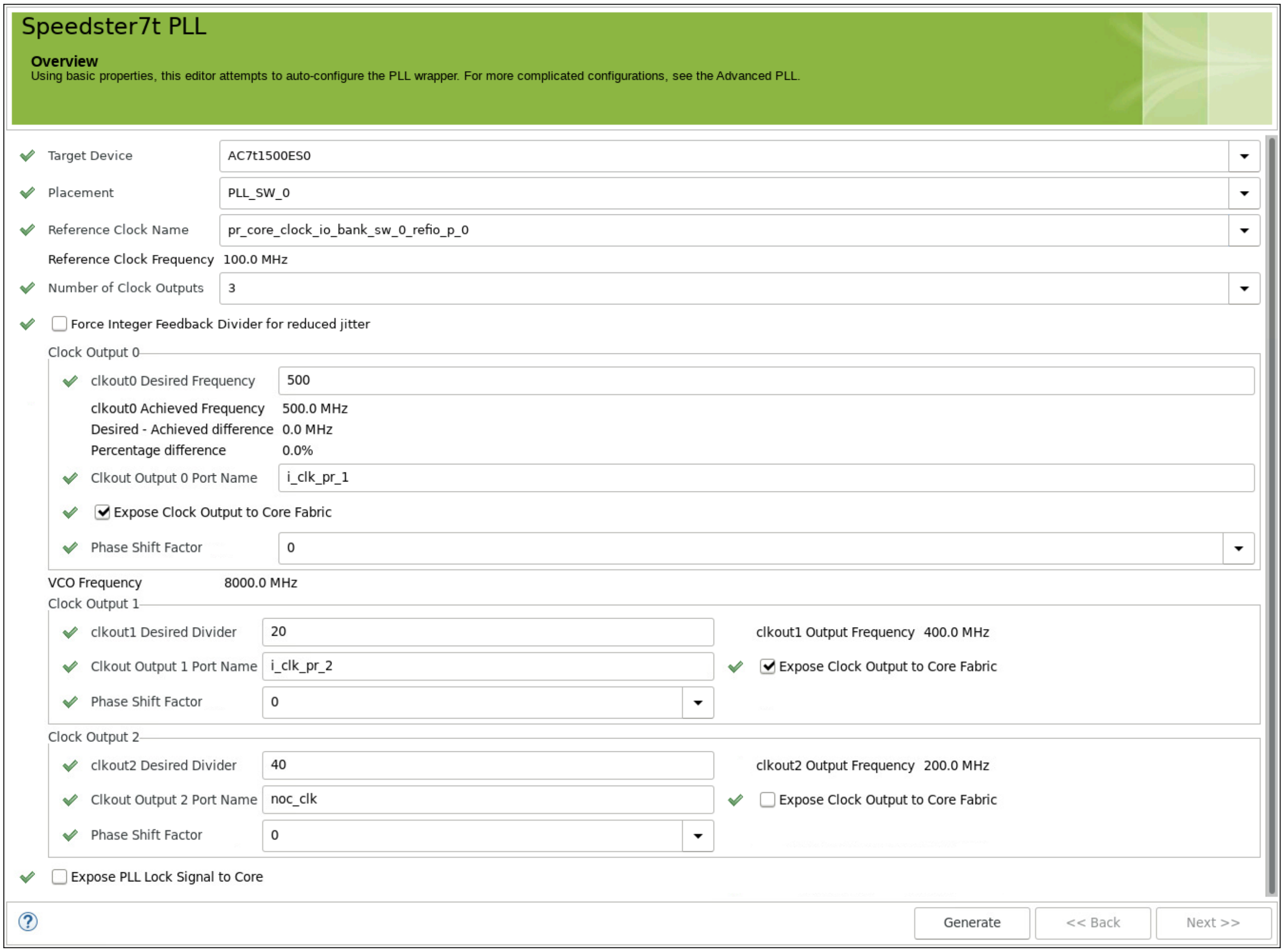


Figure 240: ***PLL IP Configuration Example***

6. In the **IP Libraries** view, right-click the **NoC** IP and select **New IP Configuration**.
7. Configure the 2D NoC IP for the PR core using the options shown in the following image:

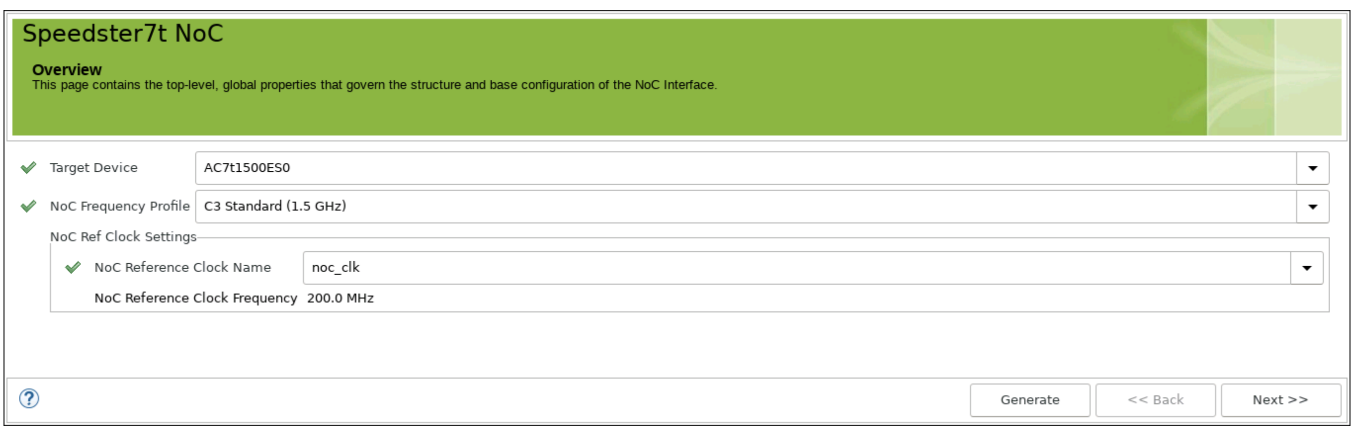


Figure 241: ***2D NoC IP Configuration Example***

8. In the **IP Libraries** view, right-click the **Clock I/O Bank** IP and select **New IP Configuration**.

9. Configure the Clock I/O Bank for the top-level design using the options shown in the following image:

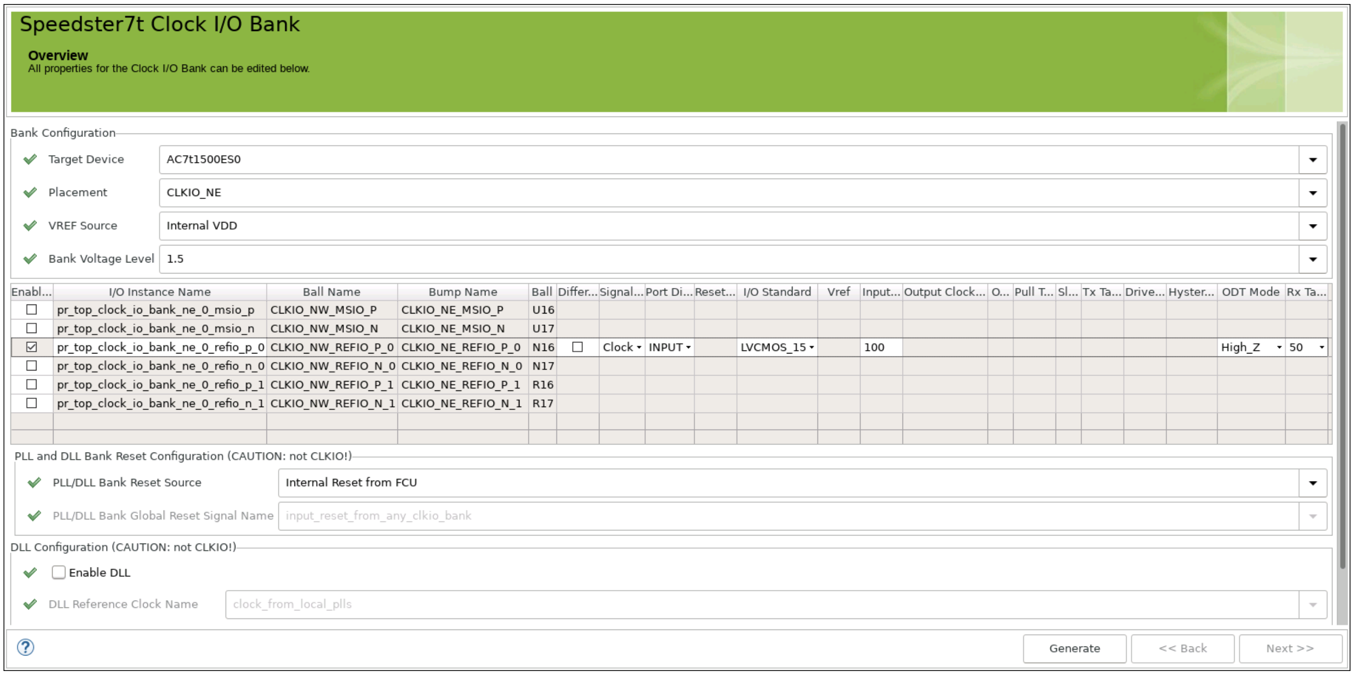


Figure 242: ***Clock I/O Bank IP Configuration Example***

10. In the **IP Libraries** view, right-click the **PLL** IP and select **New IP Configuration**.
11. Configure the PLL for the top-level design using the options shown in the following image:

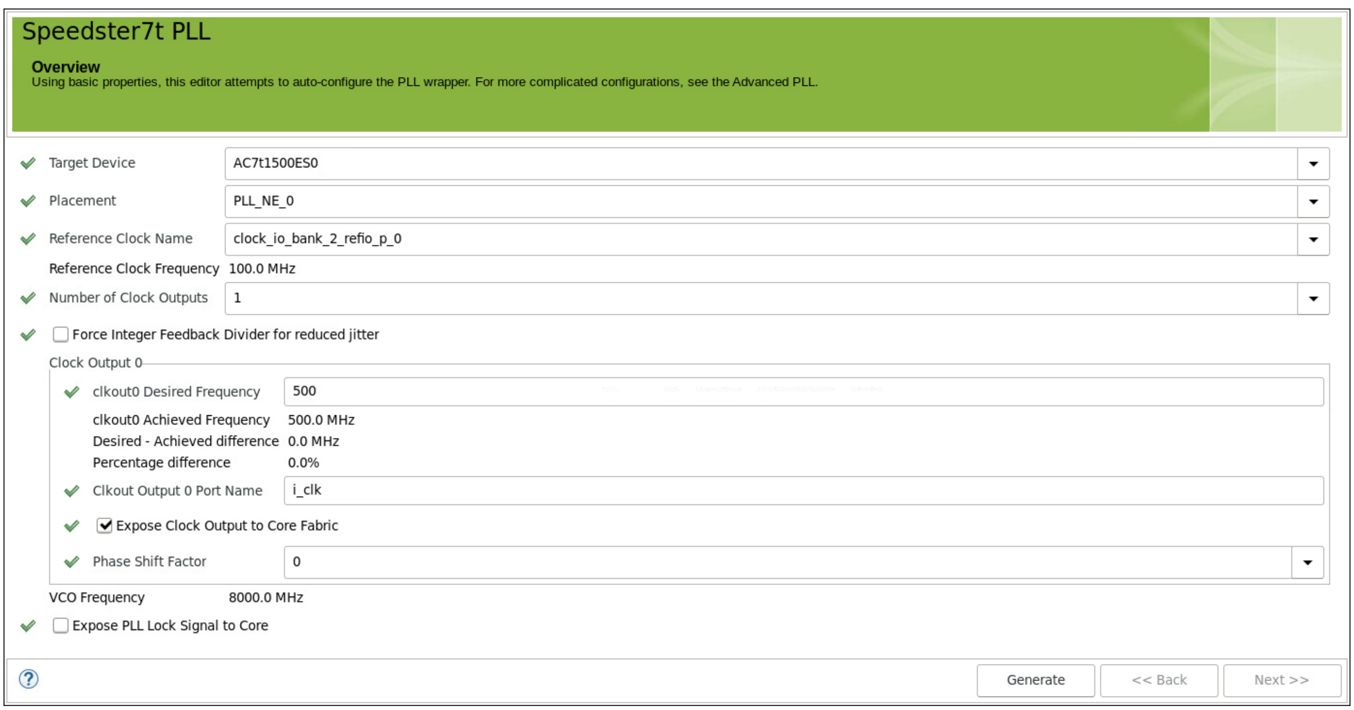


Figure 243: ***PLL IP Configuration Example***

12. In the **IP Libraries** view, right-click the **GPIO Bank** IP and select **New IP Configuration**.
13. Configure the GPIO Bank for the top-level design using the options shown in the following image:

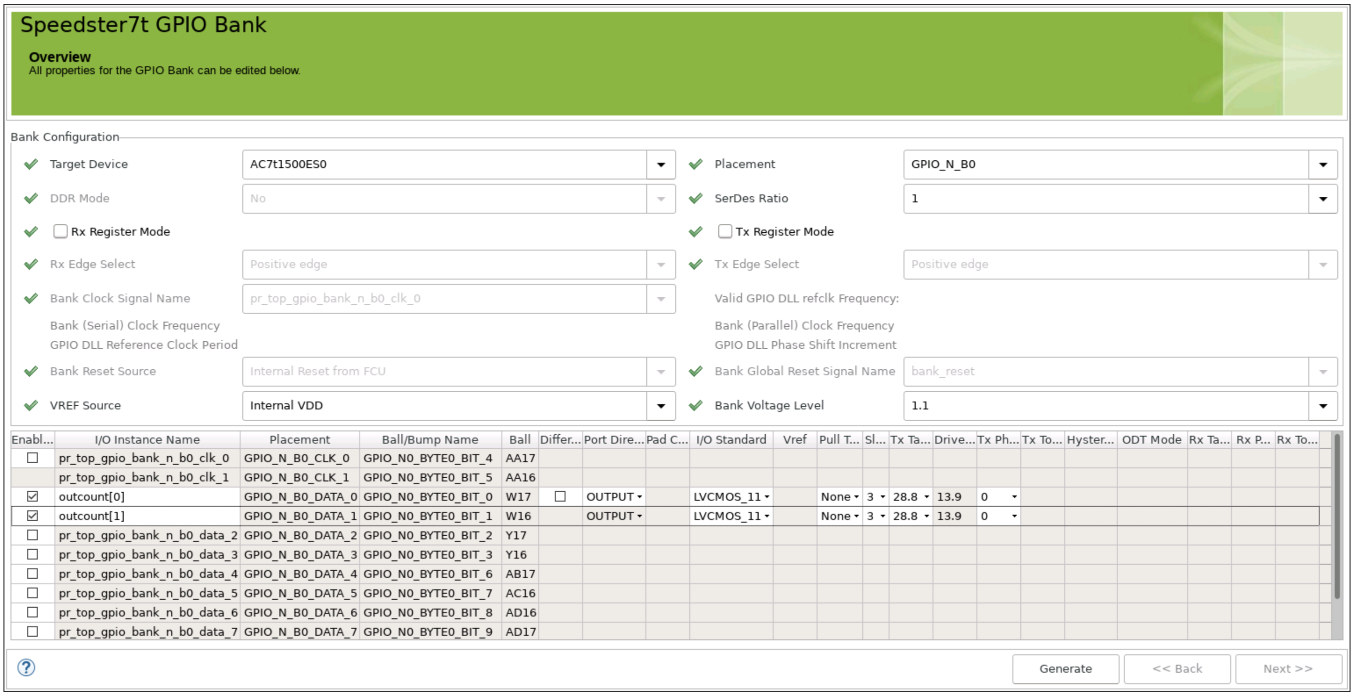


Figure 244: *GPIO Bank IP Configuration Example*

14. Click **Generate** to display the **Generate IO Ring Design Files** dialog and select **Add to active project**.

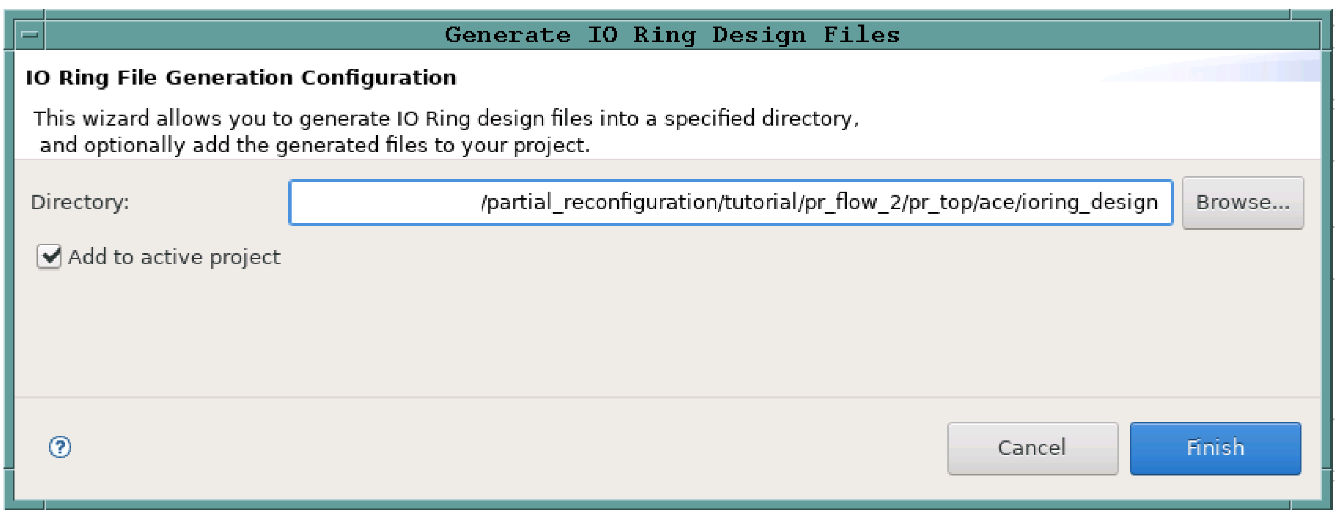


Figure 245: *Generate IO Ring Design Files Dialog Example*

15. Click **Finish** to generate the I/O ring design files.
16. When complete, the following files should be added to the ACE project:

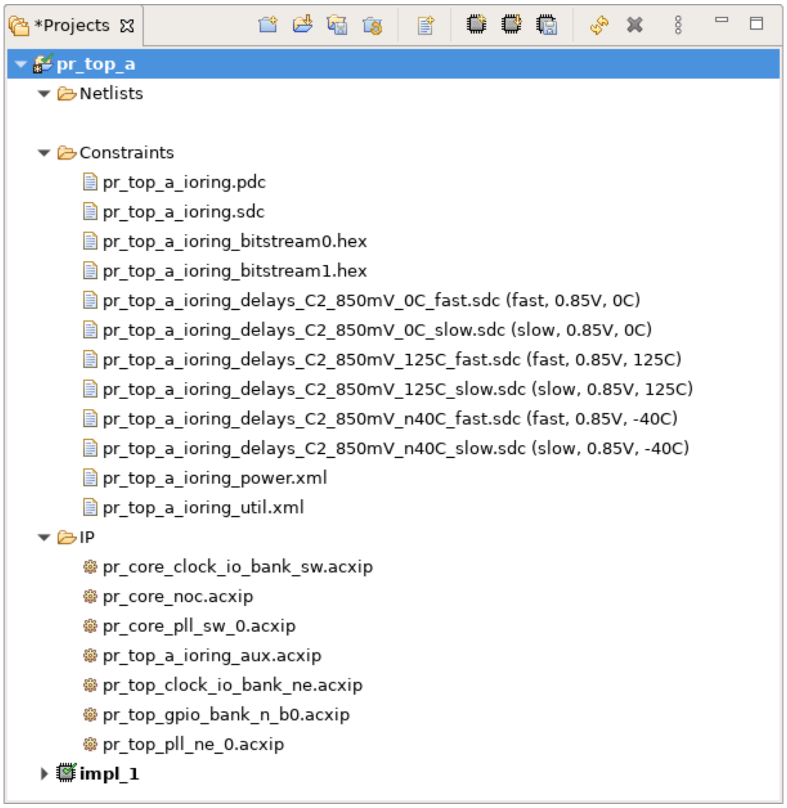


Figure 246: ***I/O Ring Design Files***

Synthesize Top-Level Design Using Synplify

1. Create a new Synplify project and add all RTL files from the `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/rtl` directory.
2. Add the `pr_top_a_timing.sdc` file from the `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top/src/constraints` directory.
3. Add the PR core 1A and PR core 2A blackboxes generated by ACE in the previous 2 steps as follows:
    - PR core 1A blackbox – `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output/blackboxes/pr_core_1a_b.v`
    - PR core 2A blackbox – `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/ace/impl_1/output/blackboxes/pr_core_2a_bb.v`
4. From the main menu, select **Project** → **Implementation Options** → **Device**.
5. Set **Technology** to **Achronix Speedster7t**.
6. Set **Part** to **AC7t1500ES0**.
7. Set **Package** to **F53**.
8. Set **Speed** to **C2**.
9. From the main menu, select **Project** → **Implementation Options** → **Implementation Results**.
10. Set **Result Base Name** to "`pr_top_a`".
11. Select **Project** → **Implementation Options** → **Verilog**.
12. Set **Top Level Module** to "`pr_top_a`".

13. Set **Include Path Order** to "<ace_install_dir>/libraries".
14. Add the AC7t1500ES0_synplify.sv file to the project from the <ace_install_dir>/libraries/device_models directory.
15. Verify that the Synplify project matches the following configuration:

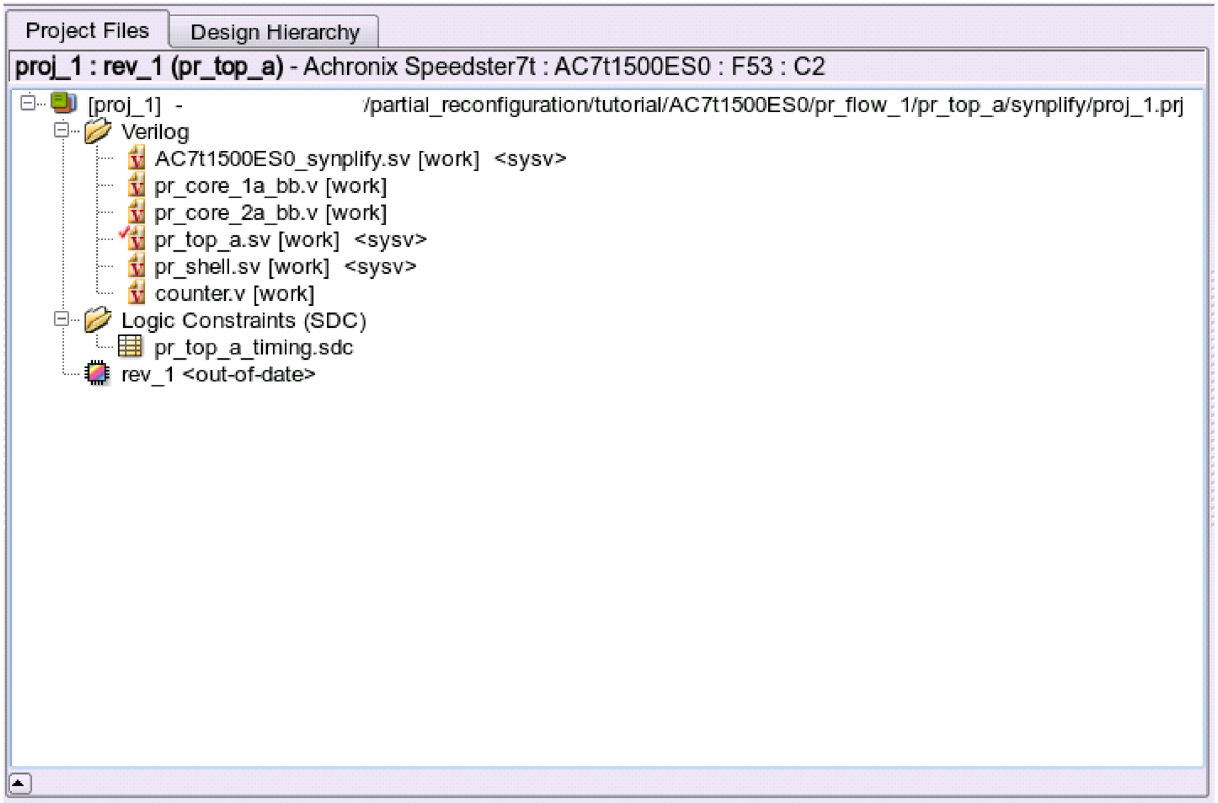


Figure 247: *Synplify Project Example*

16. Run and compile the design in Synplify.
17. When successfully completed, Synplify generates a gate-level netlist as <synplify_out_dir>/rev_1/pr_top_a.vm.

Run Top-Level Design Through Run Prepare in ACE

1. Navigate to the Projects view and click the ( ) **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the Projects view, click the ( ) **Add Source Files to a Project** button.
4. Add the following files to your project:
   - Blackbox file generated by ACE in step 1j – <ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output/blackboxes/pr_core_1a_bb.v
   - Blackbox file generated by ACE in step 2j – <ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/ace/impl_1/output/blackboxes/pr_core_2a_bb.v

- Synthesized netlist created by Synplify in step 3a – `<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/synplify/rev_1/pr_top_a.vm`

5. Add all ioring files created in step 3a to the project.
6. Ensure that the blackbox files `pr_core_1a_bb.v` and `pr_core_2a_bb.v` are both above the `pr_top_a.vm` file (click and drag the files to change the order).
7. In the **Options** view, navigate to the **Design Preparation** section and ensure that the **Export All Partitions** option is cleared and all other options are correctly set.
8. In the **Flow** view, right-click **Run Prepare** and select **Run Selected Flow Step**.
9. Ensure that the run_prepare flow step completed successfully.

Set Up Clock Pre-Routing Constraints

1. In the **Partitions** view, right-click the cell below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes** as shown in the following image:

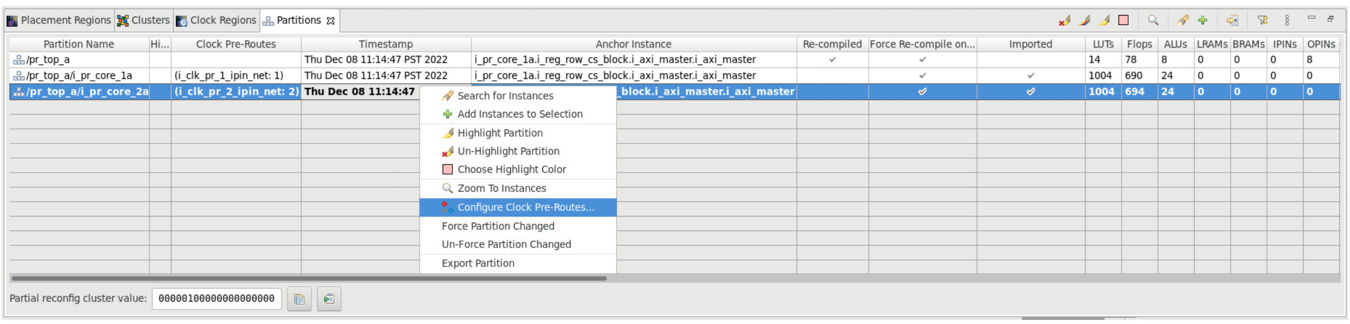


Figure 248: *Configure Clock Pre-routes Example*

2. In the **Configure Clock Pre-Routes** dialog, ensure that the clock net is pre-routed into the same track specified in step 1d:

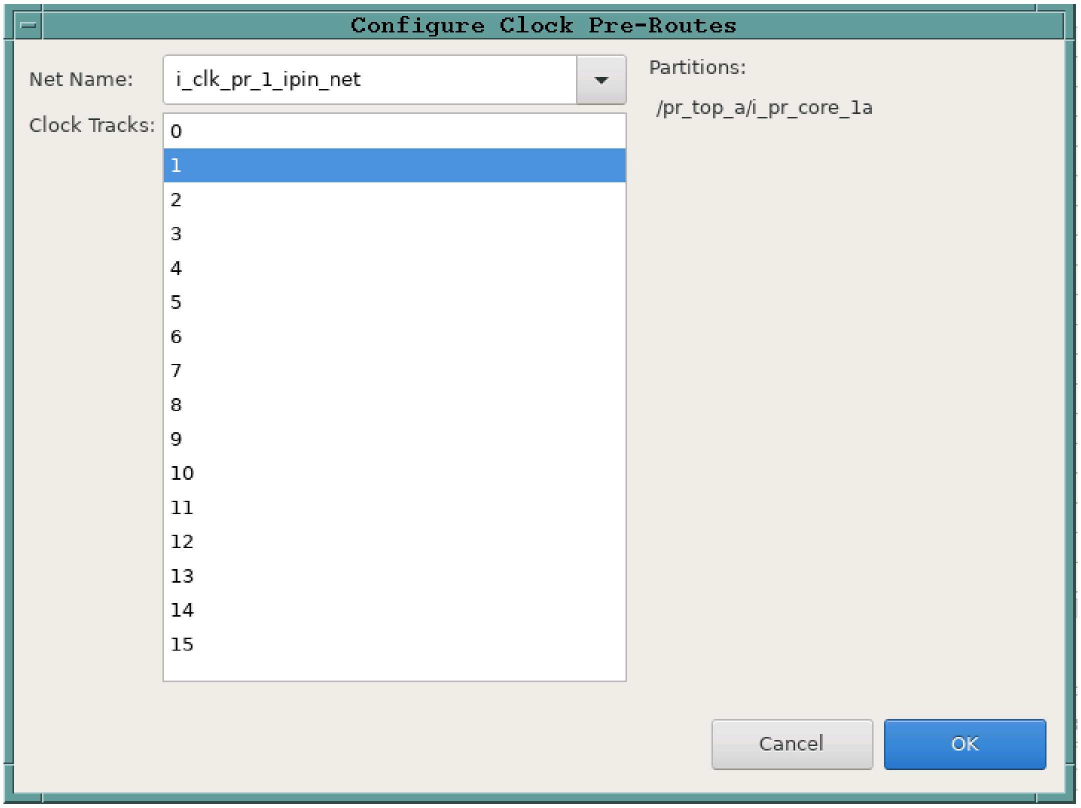


Figure 249: *Configure Clock Pre-Routes Dialog Example*

3. Click **OK**. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net 2 -partitions {/pr_top_a/i_pr_core_1a}
```

Set Up Region Constraints (Optional)

1. In the Floorplanner view, navigate to the top section and click the ( ) **Placement Region Tool** button.
2. Click and drag in the Floorplanner to draw a region bounding box for the top-level logic:

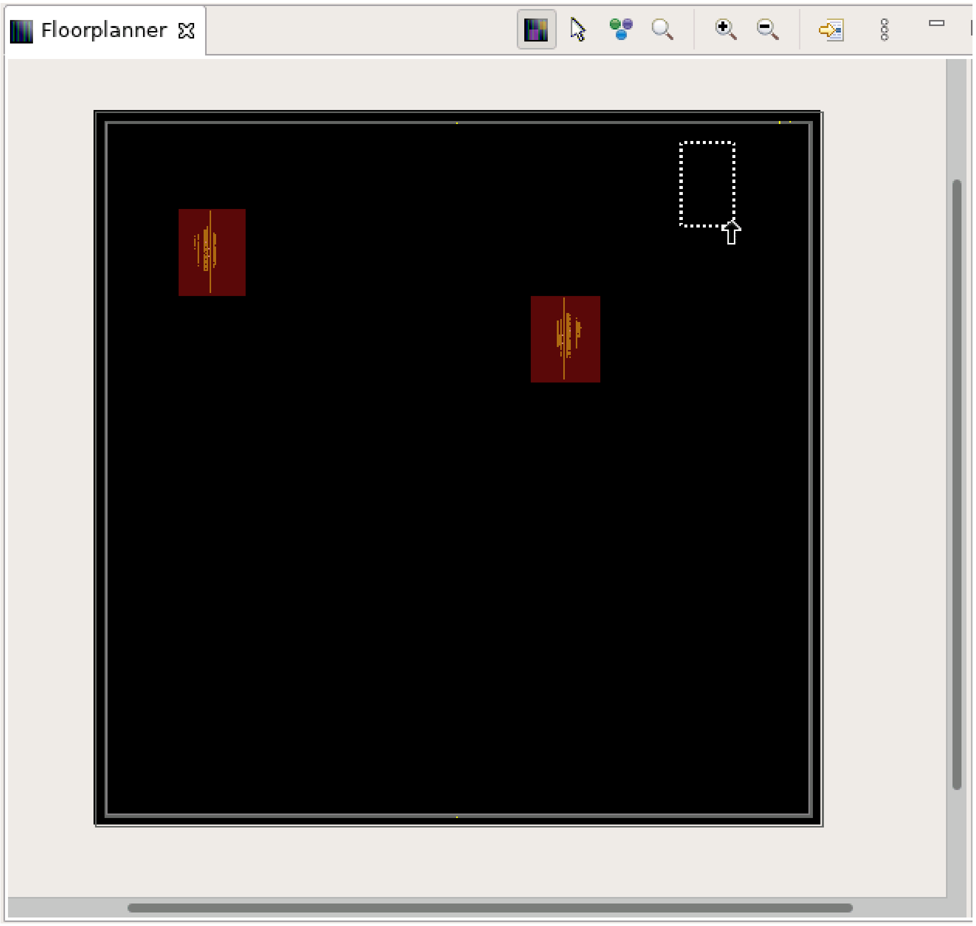


Figure 250: ***Floorplanner Region Bounding Box Example***

> **Note**
>
> ACE automatically creates a PR Zone for the imported partition.

3. In the **Create Placement Region** dialog, set **Region Alignment** to **Snap to Fabric Clusters**.
4. Set **Region Type** to **Inclusive**.
5. Clear the **Is Partial Reconfiguration Zone** checkbox.

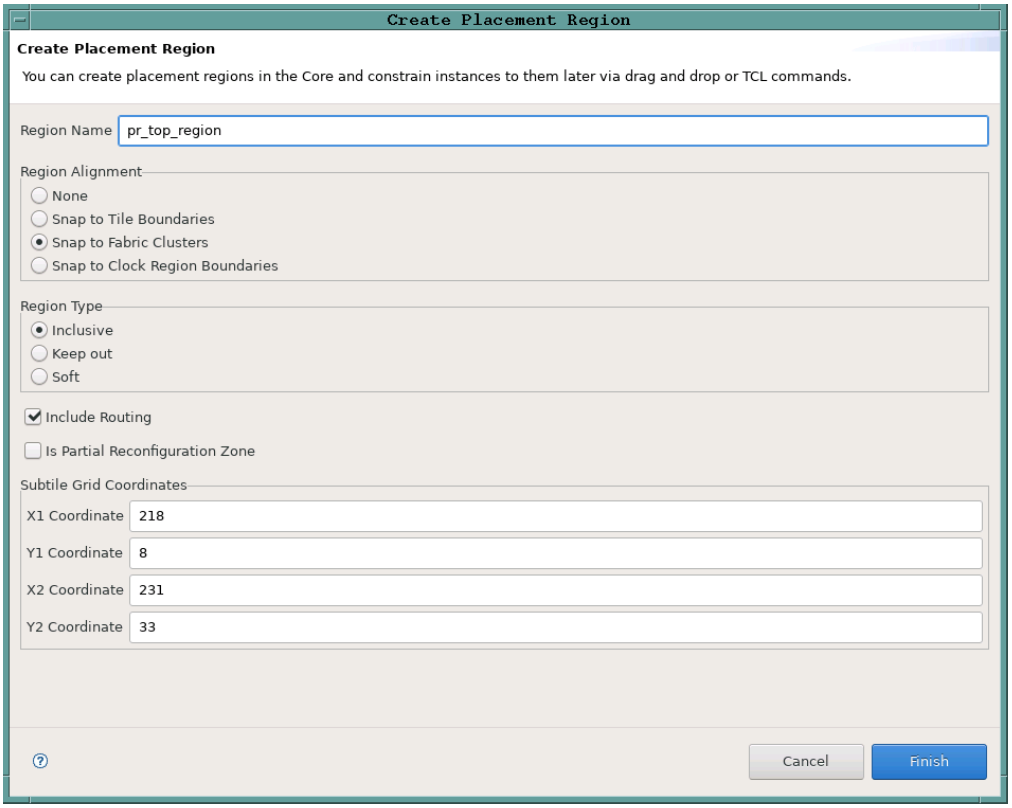


Figure 251: ***Create Placement Region Dialog Example***

6. Click **Finish**. The Tcl console displays the following:

```
create_region "pr_top_region" {218 8 231 33} -snap fabric_clusters -type inclusive -include_routing
```

7. As before, add all instances in the PR core to the region using the `add_region_find_insts` Tcl command nested with the `find` command. `i_pr_shell` is the instance name for the top-level partial reconfigurable module:

```
add_region_find_insts "pr_top_region" {find {*i_pr_shell*} -insts}
```

Run Top-Level Design Through Place-and-Route in ACE

1. In the **Flow** view, right-click **Place and Route** and select **Run Selected Flow Step**:

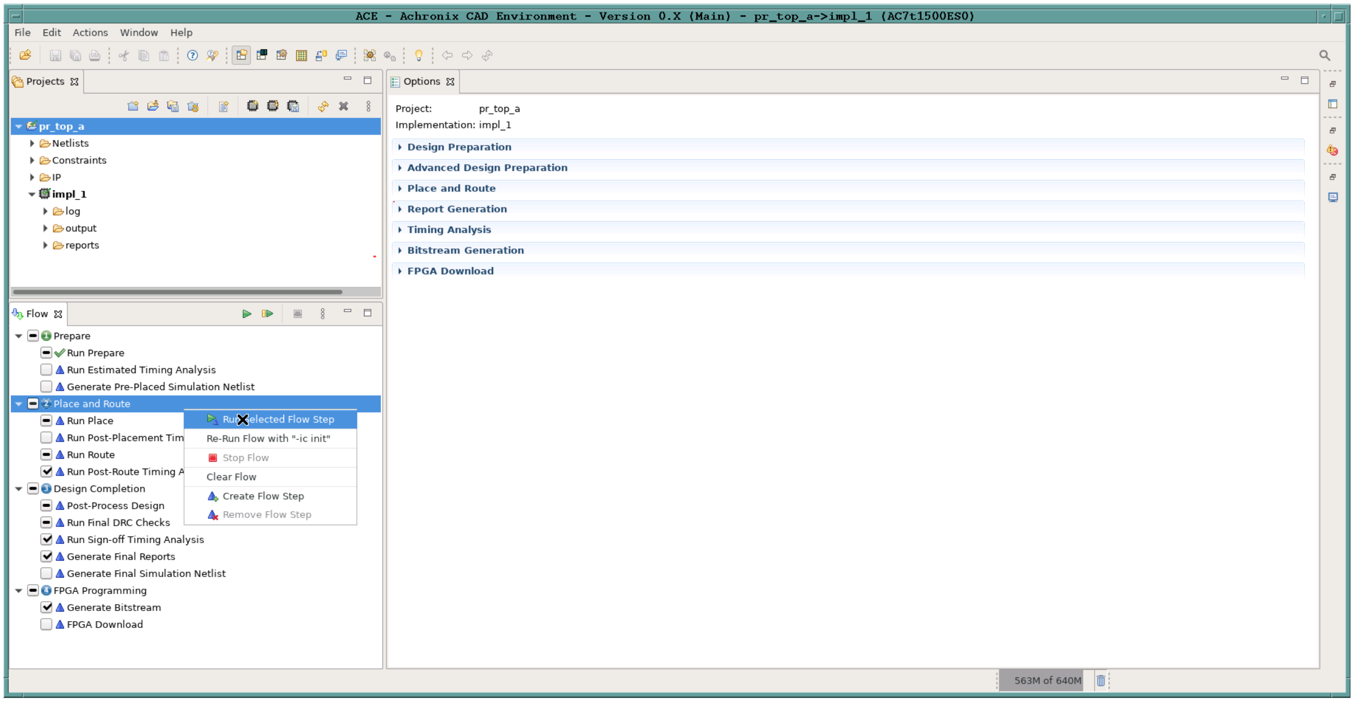


Figure 252: ***Place and Route Flow Step Example***

Run Final Sign-Off Timing Analysis in ACE

1. In the **Flow** view, right click the **Design Completion** step and select **Run Selected Flow Step**:

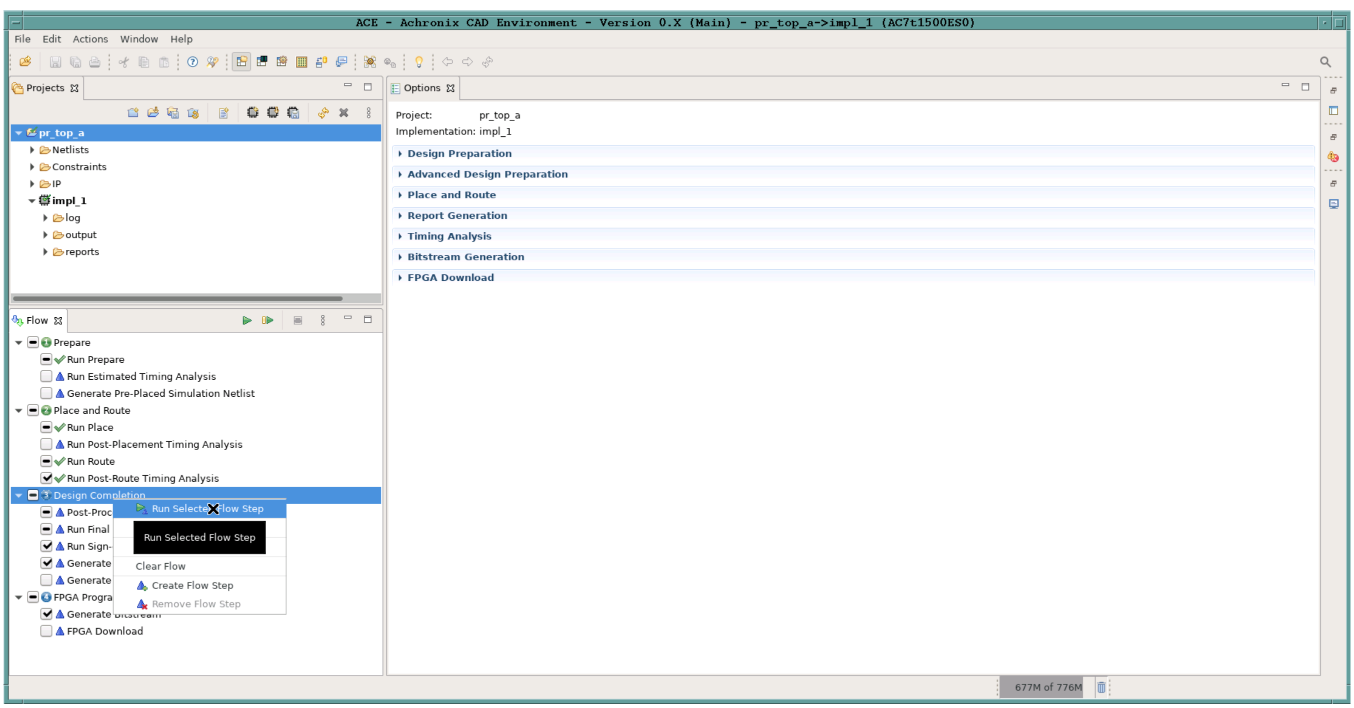


Figure 253: ***Run Design Completion Flow Step Example***

Generate Base Bitstream for Top-level Design

> **Note**
>
> The top-level base bitstream contains the top-level design plus the two imported PR cores.

1. In the "Options" view, click the **Bitstream Generation** tab.
2. In the **FCU Configuration** section, check the **Lock FCU After Programming** checkbox:

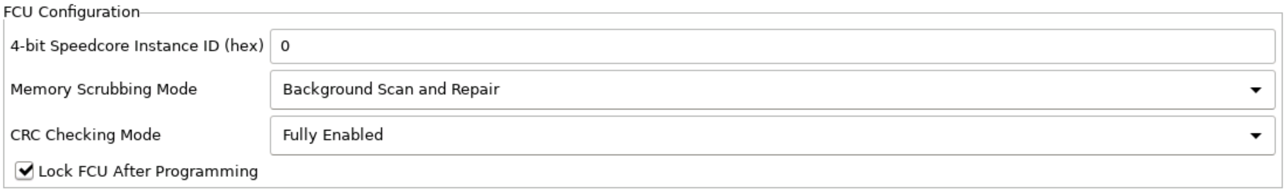


Figure 254: ***Bitstream Generation FCU Configuration Example***

3. In the Flow view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**:

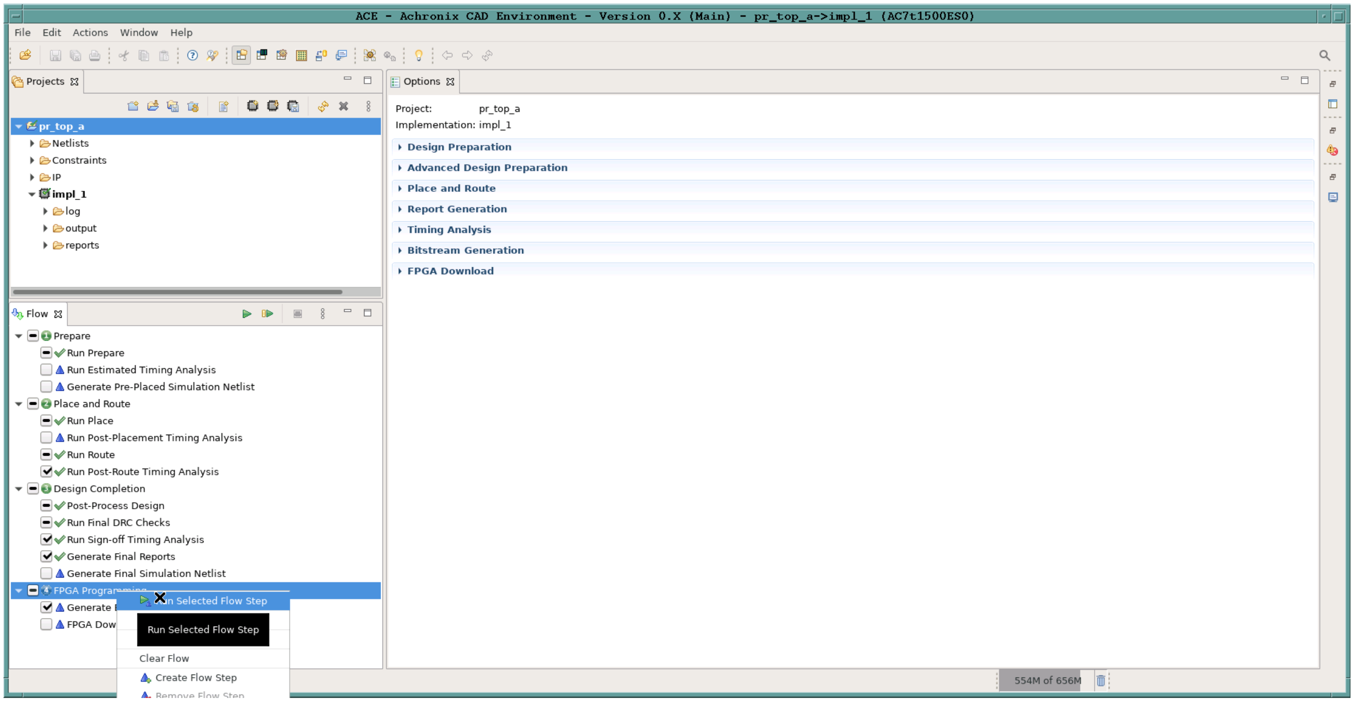


Figure 255: ***Run FPGA Programming Flow Step Example***

Save Region/Clock Pre-Routing and Partition Placement Constraints Back to PDC File

1. Repeat step 1g to save the placement region and clock pre-routing constraints to a pdc file.
2. In the **Partitions** view, click the () **Save Partition Pre-Placement** button:

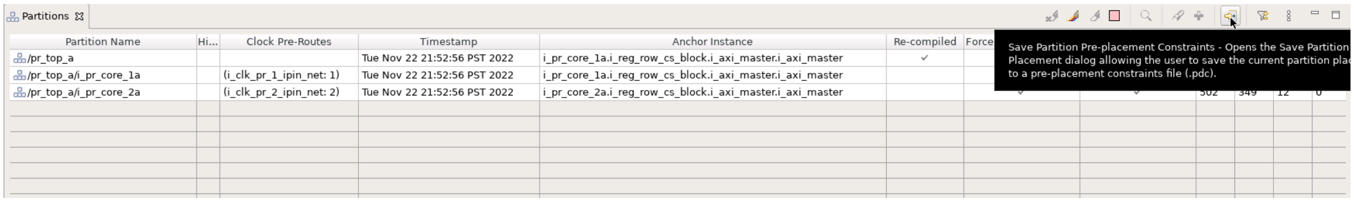


Figure 256: ***Save Partition Pre-Placement Example***

3. In the **Save Partition Pre-Placement Data** dialog, ensure that the **Save Specific List of Partitions** checkbox is cleared in order to save the partition placement constraints for each partition.

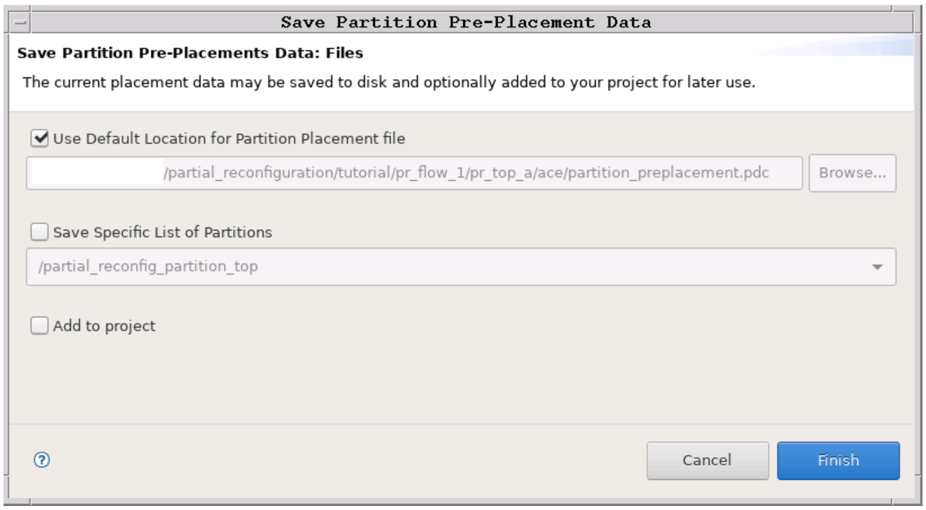


Figure 257: ***Save Partition Pre-Placement Data Dialog Example***

Generate Partial Bitstream for PR Core 1B and 2B

Export PR Core 1B as Partition

1. Repeat all steps in section 1 to export PR core 1B as a partition.
   PR core 1B is identical to PR core 1A except that the 32-bit adder increments by 2 instead of 1 at each NAP read.
   The source files for PR core 1B are located in the following directory:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b
/src/...
```

Export PR Core 2B as Partition

1. Repeat all steps in section 1 to export PR core 2B as a partition.
   PR core 2B is identical to PR core 2A except that the 32-bit subtractor decrements by 2 instead of 1 at each NAP read.
   The source files for PR core 2B are located in the following directory:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b
/src/...
```

Import PR Core 1B and 2B Into Top-Level Design

1. Create an ACE Project named "`pr_top_b`".
2. In the **Options** view, set the same options used in step 3b.
3. In the **Projects** view, click the ( ) **Add Source Files to Project** button.
4. Add the following files to the project:
   - All ioring config files generated in step 3a (the I/O in `pr_top_b` is the same as `pr_top_a` so the files can be reused)
   - `<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b/ace/impl_1/output/blackboxes/pr_core_1b_bb.v` (blackbox file generated by ACE in step 4a).
   - `<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/ace/impl_1/output/blackboxes/pr_core_2b_bb.v` (blackbox file generated by ACE in step 4b).
   - `<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_top_b/synplify/rev_1/pr_top_b.vm` (synthesized netlist created by Synplify in step 3a)
5. Ensure that the blackbox files `pr_core_1b_bb.v` and `pr_core_2b_bb.v` are both above the `pr_top_b.vm` file (drag and drop files to change the order):

Figure 258: ***PR Core Source Files Example***

6. Run the design through "Place-and-Route" (remember to configure the clock pre-routes by repeating the step 3d).
7. Ensure that the clock nets are pre-routed on the same clock track.

Generate Partial Bitstream for PR Core 1B and 2B in Top-level Design

1. Ensure that the Place-and-Route flow step completed successfully.
2. Set the bitstream cluster mask and partial reconfiguration option for PR core 1B and 2B (set the mask for both partitions by selecting both in the **Partitions** view):

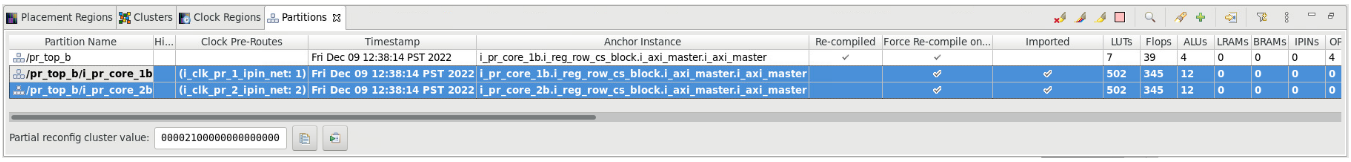


Figure 259: ***Selecting PR Core 1B and 2B Example***

3. Click the ( ) **Send Tcl Command** button to set this implementation option. The Tcl Console displays the following:

```
set_impl_option bitstream_prc_cluster_map 00002100000000000000
```

4. In the **Options** view, click the **Bitstream Generation** tab.
5. In the **Partial Reconfiguration** section, Check the **Enable Partial Reconfiguration** checkbox.
6. Ensure that the **Partial Reconfig Cluster Map** (hex) is set to the same value as in the **Placement Regions** view:

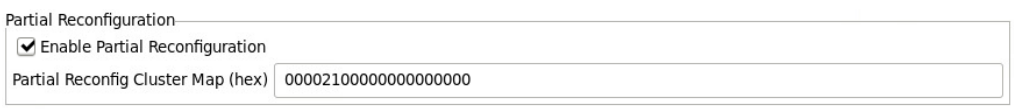


Figure 260: ***Partial Reconfiguration Settings Example***

7. In the **Options** view, click the **Bitstream Generation** tab.
8. In the **Partial Reconfiguration** section, check the **Lock FCU After Programming** checkbox:

Figure 261: ***Partial Reconfiguration Settings Example***

9. Run the **FPGA Programming** flow step to generate a single partial bitstream for PR core 1B and 2B.

Bitstream Programming Sequence

Apply Power to Board

1. Apply power to the board and ensure cables are properly connected.

Program Top-Level Base Bitstream Containing PR Core 1A and 2A

1. Run the following commands.
   Remember that the `pr_top_a` base bitstream contains PR core 1A, 2A and the top-level logic.

```
set jtag_id [jtag::get_connected_devices]
jtag::open $jtag_id
jtag::configure_scan_chain $jtag_id AC7t1500ES0 0 0 0 -single_device
jtag::ac7t1500_initialize_fcu $jtag_id -reset
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_1/pr_top_a/ace/impl_1/output/pr_top_a.hex
```

Run PR Core 1A and 2A Test Scripts

1. Run the following comnands.
   The scripts contain NAP reads from PR core 1A and 2A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/test_scripts
/read_pr_core_1a.tcl
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/test_scripts
/read_pr_core_2a.tcl
```

Program Partial Bitstream for PR Core 1B and 2B

1. Run the following command.
   PR core 1B and 2B can be partially reconfigured onto the board at the same time since a single partial bitstream was generated for both PR core 1B and 2B in step 4d.

```
#Program partial bitstream
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_1/pr_top_b/ace/impl_1/output/pr_top_b.hex
```

Run PR Core 1B and 2B Test Scripts

1. Run the following commands.
   The scripts contain NAP reads from PR core 1B and 2B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/test_scripts
/read_pr_core_1b.tcl
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/test_scripts
/read_pr_core_2b.tcl
```

PR Flow 2: Multi-Project PR Flow using Keep Out regions

Introduction

The Multi-project PR flow using keepout regions is an alternate method to create designs that can be partially reconfigured.

This flow involves using separate ACE projects to generate separate base and partial bitstreams for the top-level design and each PR core. On silicon, the base bitstream is programmed first. Partial bitstreams can be programmed subsequently. Using this flow, the user design is stitched together at a bitstream level. There is no integration in ACE between the the top-level design and each PR core since the designs are stitched at the bitstream level. As such, this flow has a few pitfalls and can only be recommended for advanced users.

Advantages of Using This PR Flow

- Very fast turn-around time in that the top-level design bypasses place-and-route of the PR cores requiring only a keep-out region
- Enables integrating third party accelerator cores at the bitstream level while not requiring any knowledge of the logic
- Instead of simulating the entire design together, partial bitstreams can be run/tested on the Achronix Virtual Lab which has a faster turn-around time than simulation

Possible Issues Encountered When Using This PR Flow

- The bitstream might not be stitched together correctly at the bitstream level if any of the ACE projects are configured incorrectly as in the following scenarios:
  1. In the base bitstream, clock nets to be used in the partial bitstream are not pre-routed. The clock signal is not correctly connected to the PR cores after partial reconfiguration.
  2. Clock nets are pre-routed on different clock tracks in the partial and base bitstreams. The clock signals are incorrectly connected to the PR cores after partial reconfiguration.
  3. In the base bitstream, keep-out regions are not set on clusters to be used by PR cores. The top-level logic might be lost after partial reconfiguration.
- The design might not meet timing on paths between the top-level and PR core after the partial bitstream is programmed since timing analysis is done separately in each ACE project.

> **Note**
>
> The timing delays on the east side are different than those on the west side of the AC7t1500ES0 device.

- Simulating the entire design within an ACE project is difficult as no integration exists between separate ACE projects. It is not impossible but requires a lot of manual work.

To avoid these issues, consider using the multi-project flow with partition export/import (PR flow 1) instead. In that flow, ACE can integrate separate projects created for both the top-level design and for each PR core by taking advantage of the partition export/import feature. This feature enables ACE to run checks to ensure that the top-level design and each PR core can be stitched together correctly. Timing can be closed and simulation run on the integrated design.

High-level design flow

The bottom-up flow is recommended for PR flow 2. This involves generating the partial bitstreams for each of the PR cores before generating the full bitstream for the top-level design. In the the top-level design, keep-out regions and clock pre-routes on each cluster where the PR cores are partially reconfigured, must be set. As such, it is easier to finish floor planning and generating the partial bitstream for each PR core before working on the top-level design used to create the full bitstream.

Please follow this high-level design flow carefully:

1. Partial Bitstream Generation for PR Core 1A (see page 479)
   a. Synthesize PR Core 1A Using Synplify (see page 479)
   b. Run PR Core 1A Through Run Prepare in ACE (see page 481)
   c. Set Up Region Constraints for PR Core 1A (see page 482)
   d. Set Up Clock Pre-Routing Constraints for PR Core 1A (see page 485)
   e. Run PR Core 1A Through Place-and-Route in ACE (see page 487)

Partial Bitstream Generation for PR Core 1A

Synthesize PR Core 1A Using Synplify

1. Navigate to the following directory:
   `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src`

> **Note**
>
> `<ace_install_dir>` is the directory path where ACE was installed.

Below this directory should be the following two directories:

- `rtl`
- `constraints`

2. In the `rtl` directory, create a new Synplify project and add all of the RTL files.
3. Add the `pr_core_1a_timing.sdc` file to the `constraints` directory.
4. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
5. Set **Part** to **AC7t1500ES0**.
6. Set **Package** to **F53**.
7. Set **Speed** to **C2**.
8. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_core_1a_top**.
9. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_core_1a_top**.
10. Set **Include Path Order** to **<ace_install_dir>/libraries**.
11. In the `<ace_install_dir>/libraries/device_models` directory, add the `AC7t1500ES0_synplify.sv` file to the project

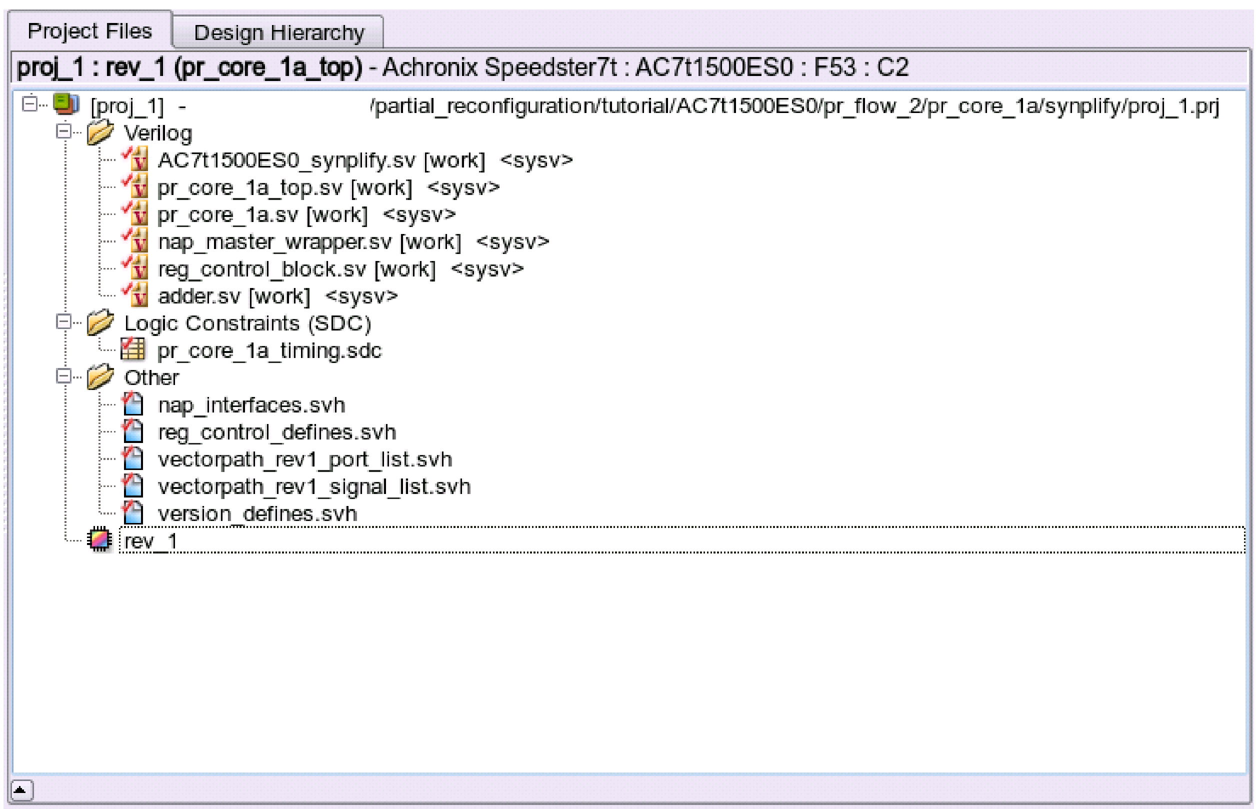


Figure 262: *Project Example*

12. Run and compile the design using Synplify.
13. After successful completion, Synplify generates a gate-level netlist under <synplify_out_dir>/rev_1/pr_core_1a_top.vm.

Run PR Core 1A Through Run Prepare in ACE

1. In the **Projects** view, click the ( ) **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the **Projects** view, click the ( ) **Add Source Files to a Project** button.
4. Add the following files to your ACE project:
    - <synplify_out_dir>/rev_1/pr_core_1a_top.vm. (synthesized netlist created by Synplify in step 1a)
    - <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/ioring_design/pr_core_1a_top_ioring.pdc

    > **Note**
    >
    > This file is required to generate a bitstream because all top-level core fabric pins need to be pre-placed.
    >
    > For partial bitstream projects, either create a dummy .pdc file for the top-level pins, or, use the .pdc file generated from the top-level design through the I/O Designer.

    - <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/ioring_design/pr_core_1a_top_ioring.sdc
5. In the **Options** view, navigate to the **Design Preparation** tab and make sure all options are correctly set.
6. Copy the option values shown in the following example.
7. In the **Flow** view, right click **Run Prepare** and select **Run Selected Flow Step**.

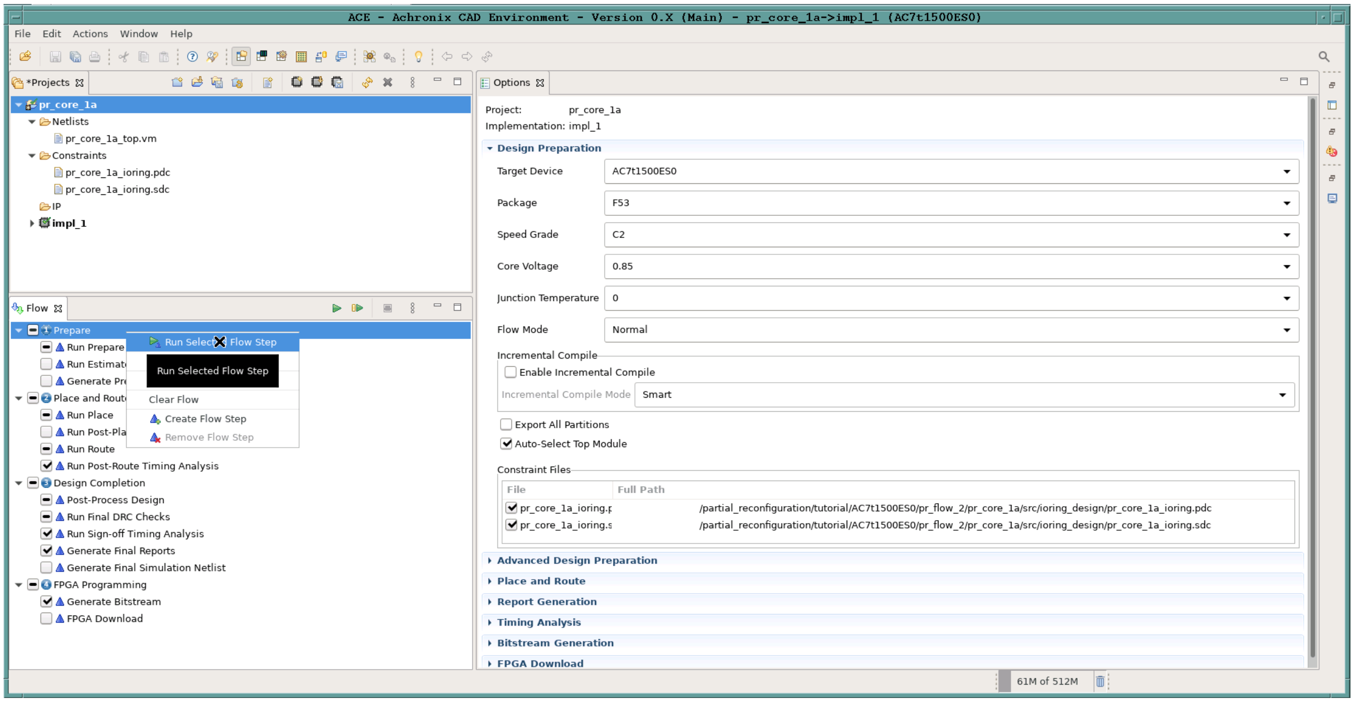


Figure 263: ***Run Prepare Flow Step Example***

Set Up Region Constraints for PR Core 1A

1. Ensure that the run_prepare flow step has completed successfully.
2. In the **Floorplanner** view, click the () **Placement Region Tool** button.
3. Click and drag on the Floorplanner to draw a PR Zone for the PR Core.

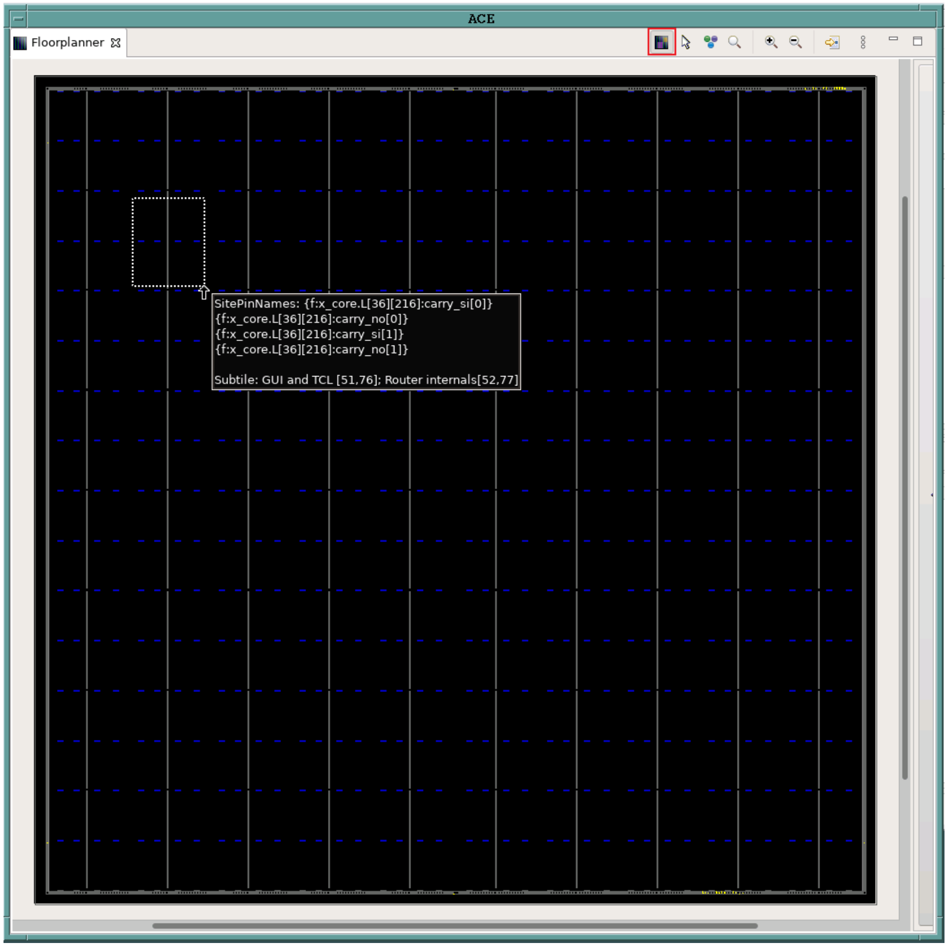


Figure 264: *Creating a Placement Region Zone Example*

4. After releasing the left mouse button, the **Create Placement Region** dialogue appears.
5. In **Region Name**, enter a name for the PR zone
6. Set **Region Alignment** to **Snap to Fabric Clusters** (a fabric cluster represents the lowest level of granularity at which the core fabric can be partially reprogrammed).
7. Set **Region Type** to **Inclusive** (ensures that all instances are constrained to the region).
8. Select **Include Routing** (ensures that all routed nets are constrained to the region).
9. Select **Is Partial Reconfiguration Zone** (must be selected to indicate this region is a PR Zone).

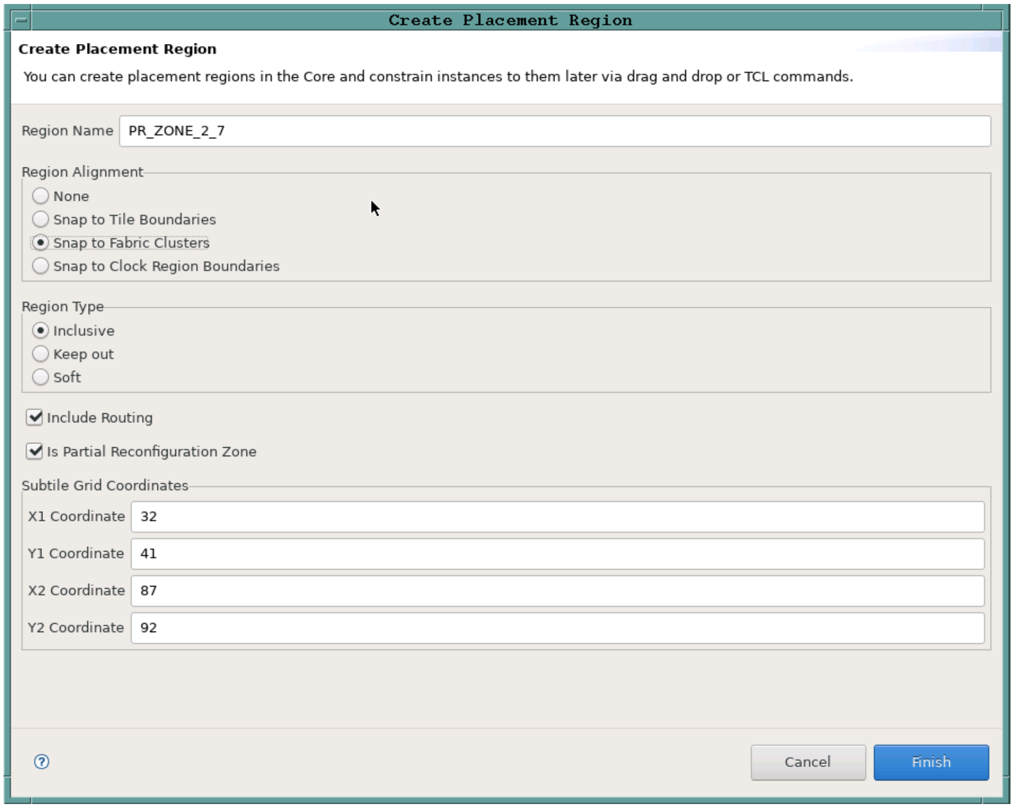


Figure 265: *Create Placement Region Dialog Example*

10. Click **Finish**. The Tcl console displays the following:

```
create_region "PR_ZONE_2_7" {32 41 48 77} -snap fabric_clusters -type inclusive -include_routing -
pr_zone
```

11. Add all the instances in the PR Core to the region. A quick and recommended way of doing so is to use the `add_region_find_insts` Tcl command nested with the `find` Tcl command. Run the following example in the Tcl console to add all instances in PR Core 1A to the PR zone (`i_pr_core_1a` is the instance name for the top-level partial reconfigurable module).
12. In the GUI, drag the PR core 1A module in the netlist browser view to the PR_ZONE_2_7 to issue this Tcl command:

```
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

13. In the **Floorplanner** tab, the image should resemble the following (the red box represents the bounding box of the PR zone just created):

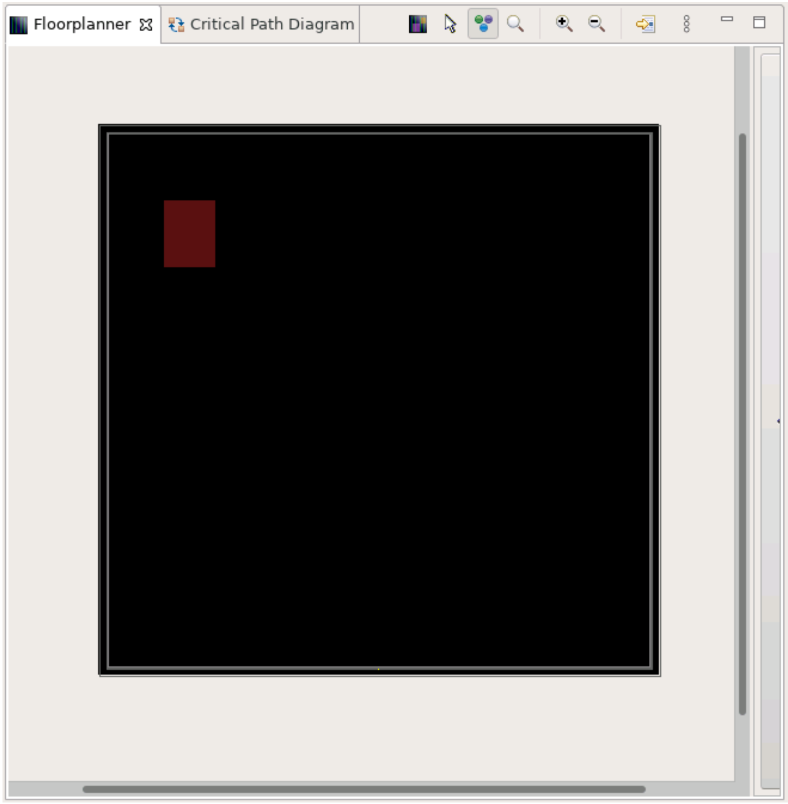


Figure 266: ***PR Zone Bounding Box Example***

Set Up Clock Pre-Routing Constraints for PR Core 1A

There are several different methods for setting up clock pre-routing. For PR flow 2, the easiest way to configure clock pre-routes is to use the **Placement Regions** view.

1. In the **Placement Regions** view, right-click the box below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes** as shown:

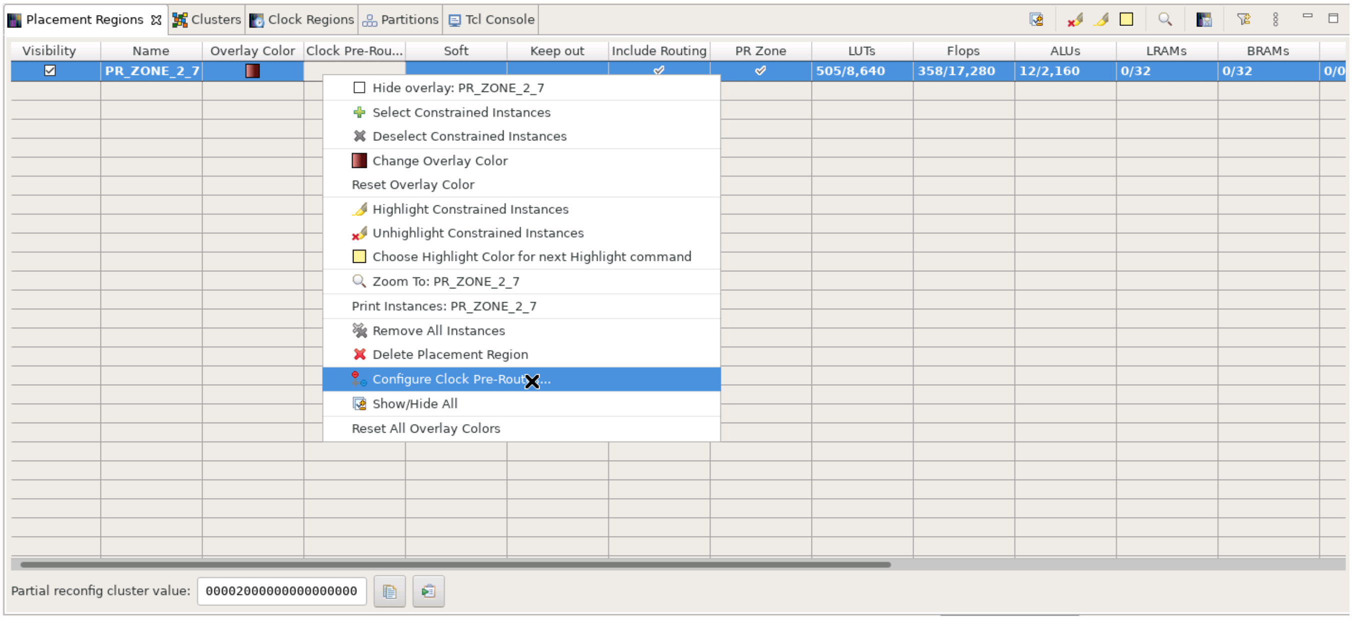


Figure 267: *Configure Clock Pre-Routes Example*

2. The **Configure Clock Pre-Routes** dialog appears with the `i_clk_pr_1_ipin_net` signal that drives PR core 1 clock pins pre-routed on clock track 1.

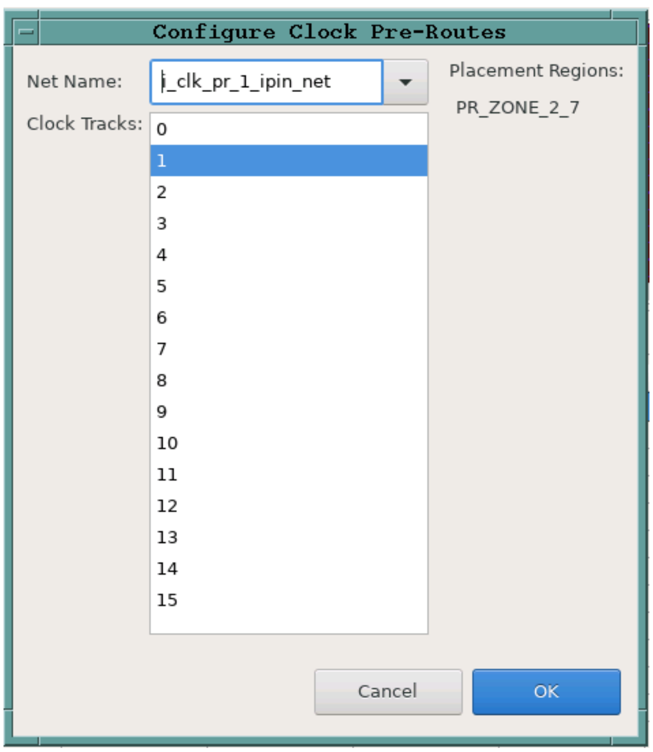


Figure 268: *Configure Clock Pre-Routes Dialog Example*

3. Click **OK**. The Tcl console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -placement_regions { PR_ZONE_2_7 }
```

> **Note**
>
> Each clock net used in the top-level design must be routed on a different clock track.

Run PR Core 1A Through Place-and-Route in ACE

1. In the **Flow** view, right-click the **Place and Route** step and select **Run Selected Flow Step**.

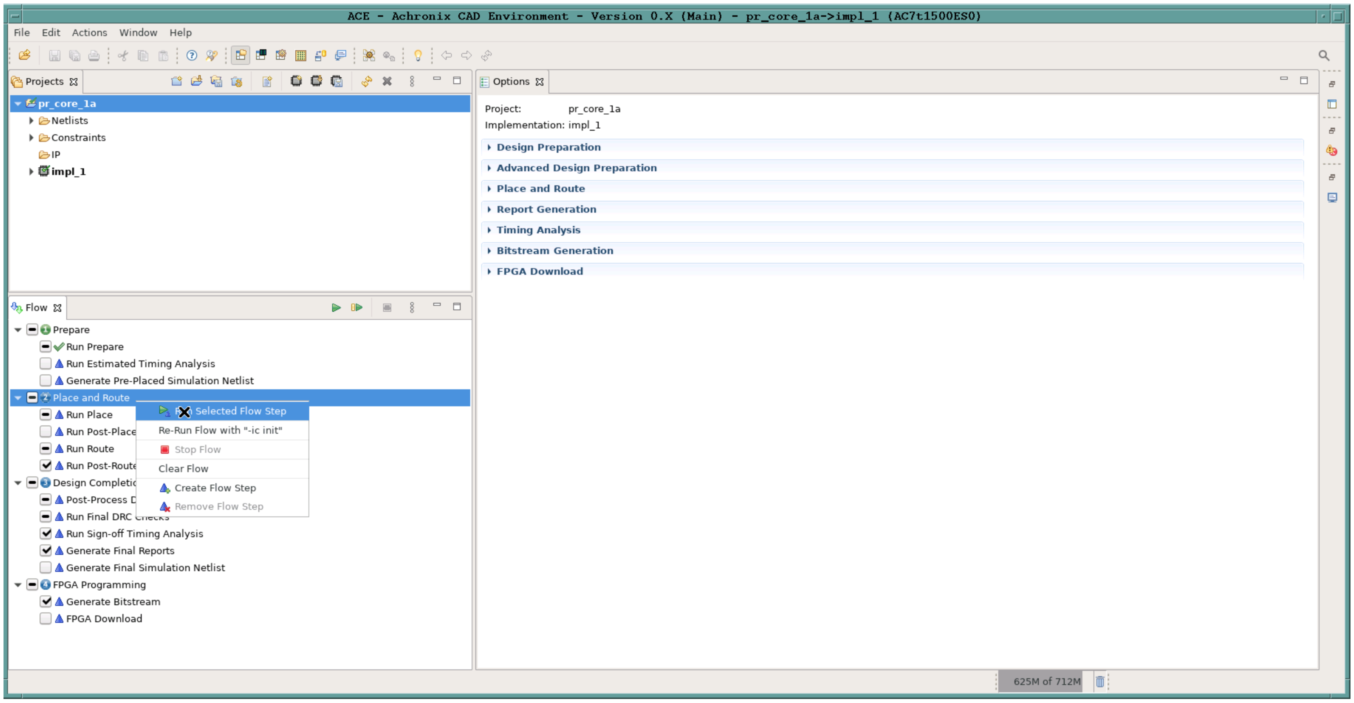


Figure 269: ***Run Place and Route Flow Step Example***

Run Final Sign-Off Timing Analysis for PR Core 1A

1. In the **Flow** view, right-click **Design Completion** and select **Run Selected Flow Step**.

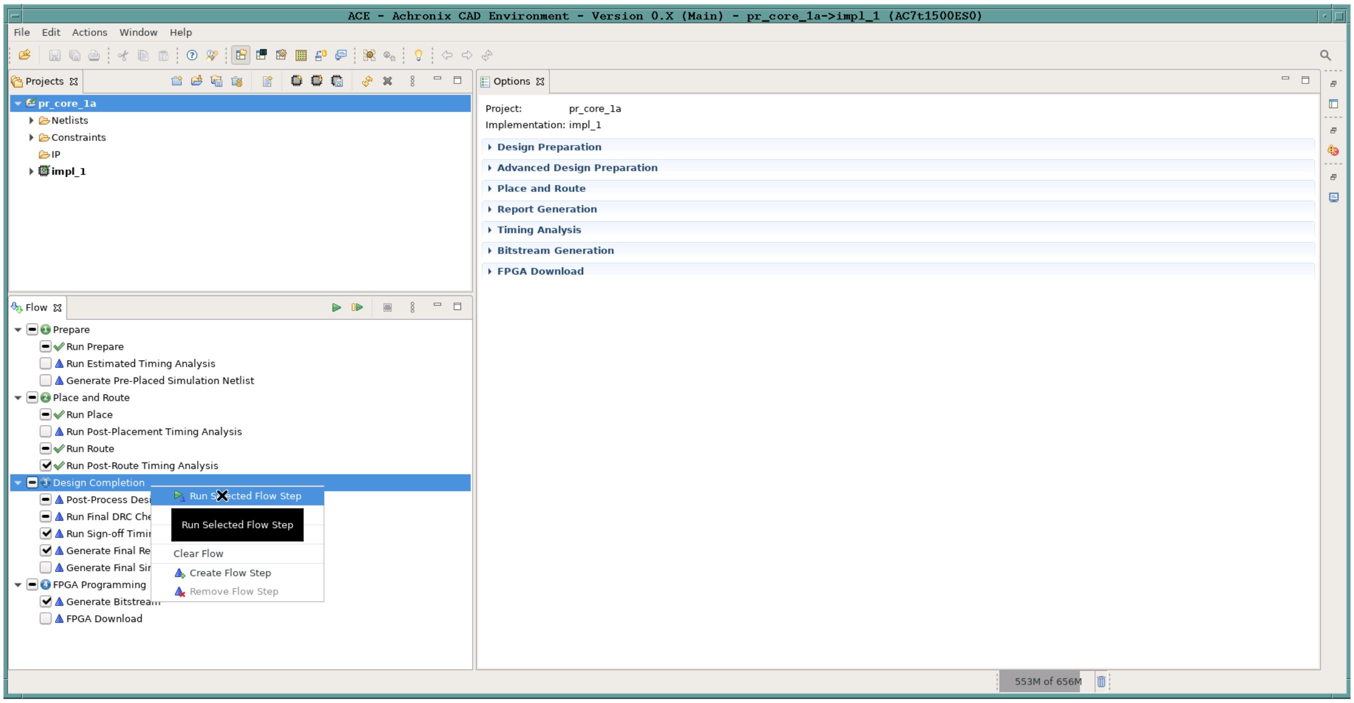


Figure 270: ***Run Design Completion Flow Step Example***

Set Bitstream Cluster Mask and Partial Reconfiguration Option for PR Core 1A

1. Ensure that the **Design Completion** flow step has run successfully.
2. In the **Placement Regions** view, check the **Visibility** checkbox for the PR_ZONE_2_7 region to be partially reconfigured (the partial reconfig cluster value is computed based on the placement regions having the **Visibility** item checked.
   At the bottom of the **Placement Regions** view, in the **Partial reconfig cluster value** section, the bitstream cluster mask value appears for the PR zone region created in step 1c. This value indicates which clusters are occupied by the PR zone. When the `bitstream_prc_cluster_map` implementation option is set to this value, a partial bitstream is generated for the clusters within the PR Zone.
3. Click the ( ) **Send Tcl Command** button to set this implementation option.

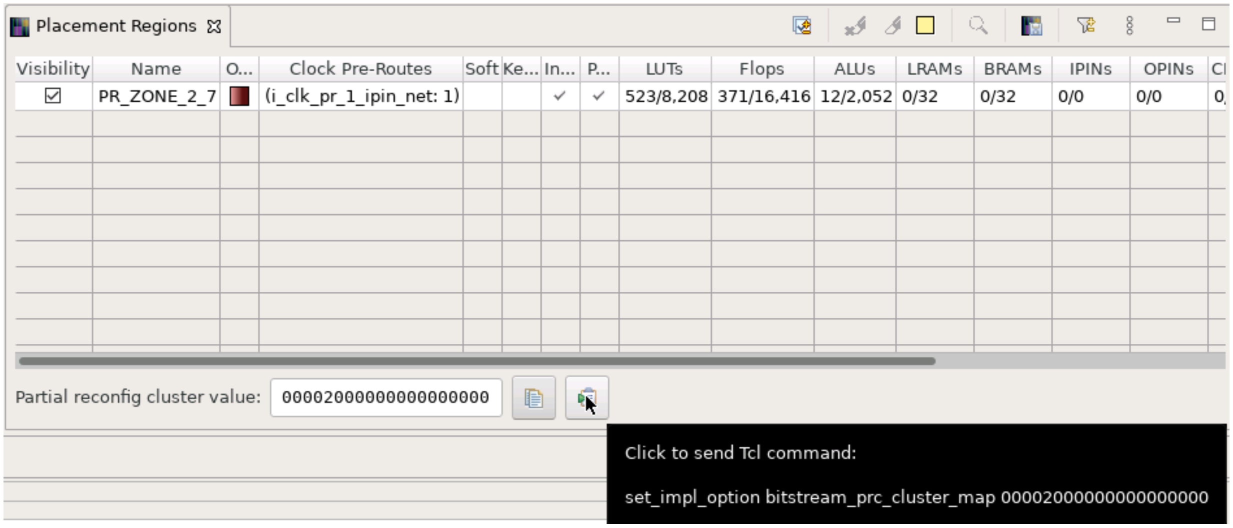


Figure 271: *Send Tcl Command Example*

4. In the **Options** view, click the **Bitstream Generation** tab.
5. In the **Partial Reconfiguration** section, check the **Enable Partial Reconfiguration** checkbox and ensure that the **Partial Reconfig Cluster Map (hex)** is set to the value from the **Placement Regions** view:

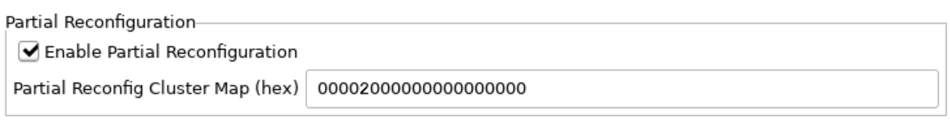


Figure 272: *Partial Reconfiguration Settings Example*

6. In the **FCU Reconfiguration** section, ensure that the **Lock FCU After Programming** item is checked:

Figure 273: *FCU Configuration Settings Example*

Generate Partial Bitstream for PR Core 1A

1. In the **Flow** view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**.

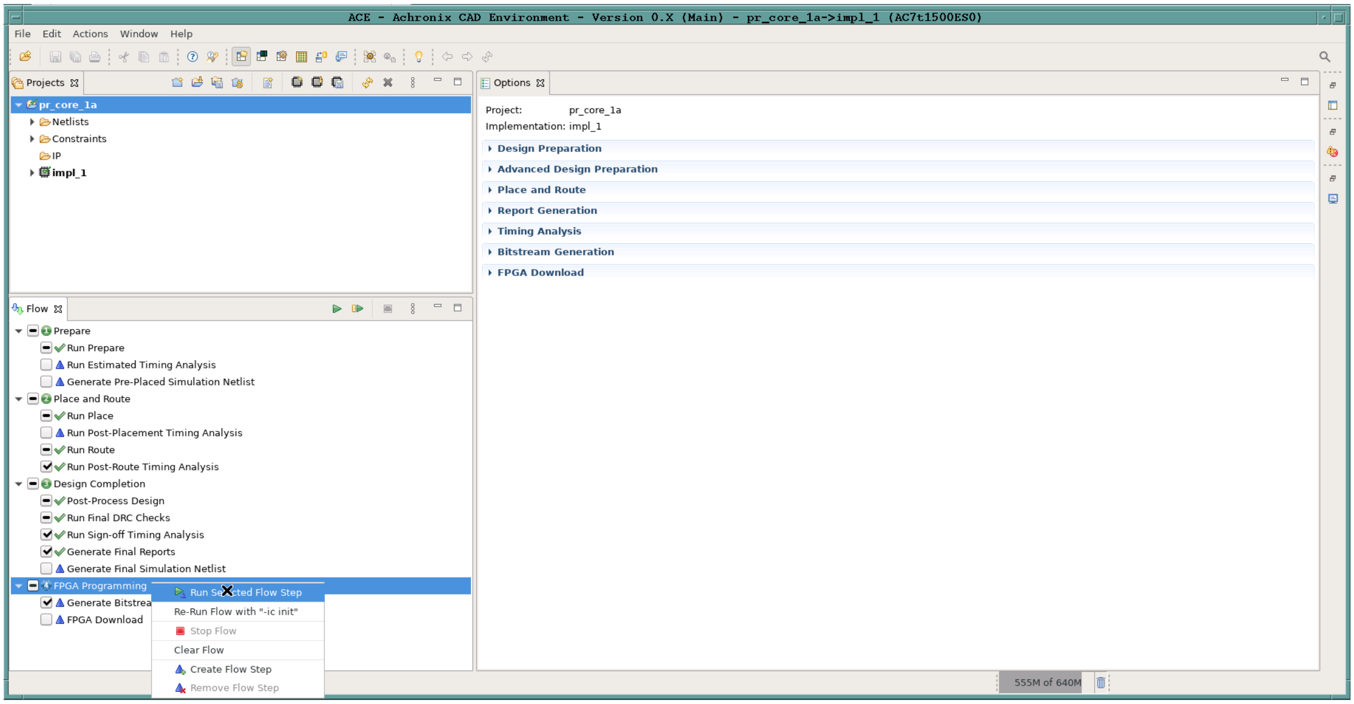


Figure 274: ***Run FPGA Programming Flow Step Example***

Save Region and Clock Pre-Routing Constraints to PDC File

Subsequent runs must use the same clock pre-routing and placement region constraints. These constraints must be saved to `.pdc` files and added to the ACE project.

> **Warning!**
>
> Adding `.pdc` files to an ACE project invalidates the flow and requires re-running the flow steps beginning with Run Prepare.

1. In the **Placement Regions View**, click the () **Save Placement Regions** button:

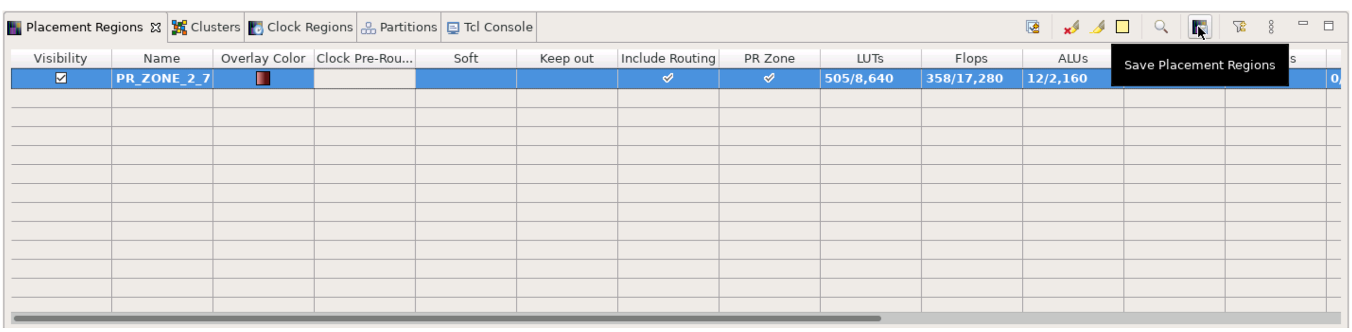


Figure 275: ***Save Placement Regions Example***

2. The **Save Placement Regions** dialog appears. Click **Finish** to save the PR zone region constraints:

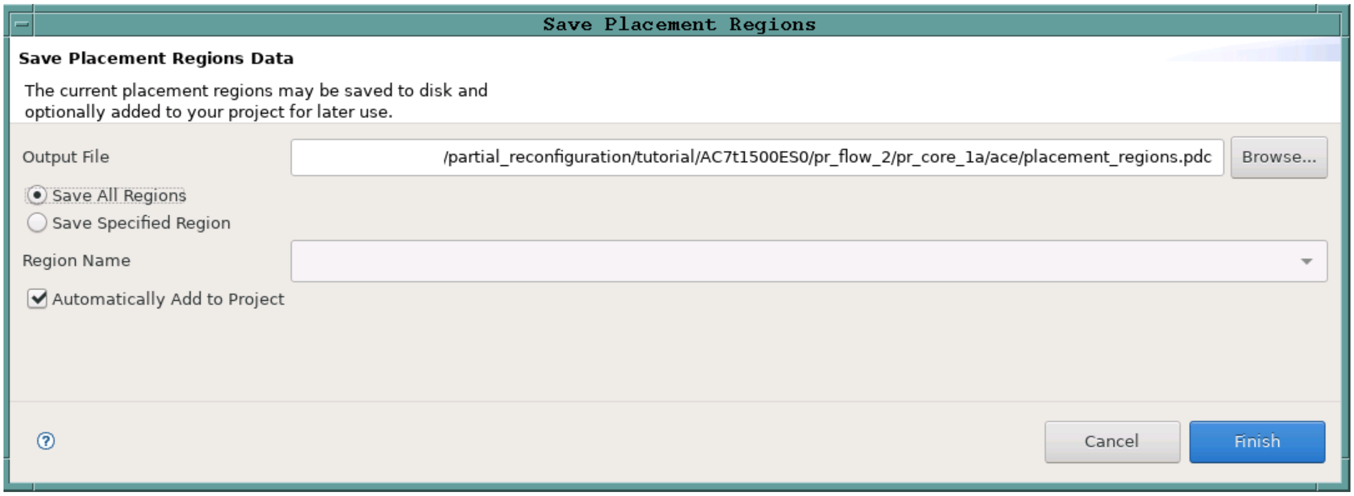


Figure 276: ***Save Placement Regions Dialog Example***

3. The Tcl console displays the following and the `placement_regions.pdc` file is created.

```
create_region "PR_ZONE_2_7" {32 41 48 77} -snap fabric_clusters -type inclusive -include_routing -
pr_zone
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

4. ACE automatically generates a clock pre-route file to the output area under `<ace_output_dir>/ace/impl_1/output/pr_core_1a_top_clock_preroutes.pdc`.

> **Note**
>
> This file should not be added to the ACE project as it is regenerated during each run. Make a copy of this file before adding it to the project.
>
> Alternately, use the `save_clock_preroute` Tcl command to generate a `.pdc` file with the clock pre-route constraints.

Generate Partial Bitstream for PR Cores 1B 2A and 2B

Generate Partial Bitstream for PR Core 1B

1. Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:
   - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1b`

Generate Partial Bitstream for PR Core 2A

1. Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:
   - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a`

Generate Partial Bitstream for PR Core 2B

1. Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:

- `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b`

Generate Full Bitstream for Top-Level Design

Create I/O Ring Configuration for Target Board

In this design, a simple clock input is driving a PLL to create clocks for the core fabric. Also, a simple GPIO interface to display LEDs is in the top-level design.

I/O Ring Configuration Steps

1. In the IP **Libraries** view, under **IO Ring**, right-click **Clock I/O Bank** and select **New IP Configuration**.

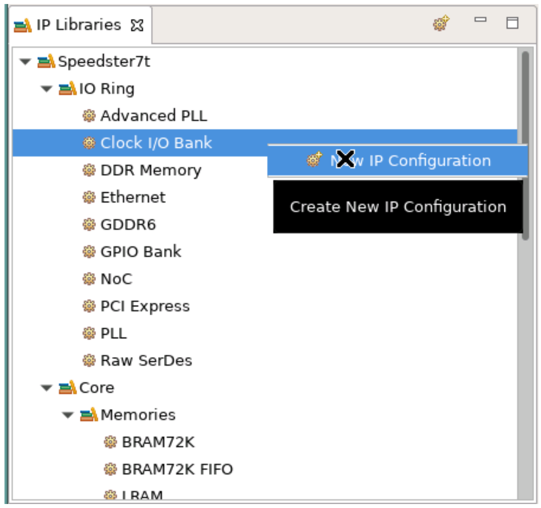


Figure 277: ***Clock I/O Bank New IP Configuration Example***

2. Configure the Clock I/O Bank for the PR Cores using the options shown in the following example:

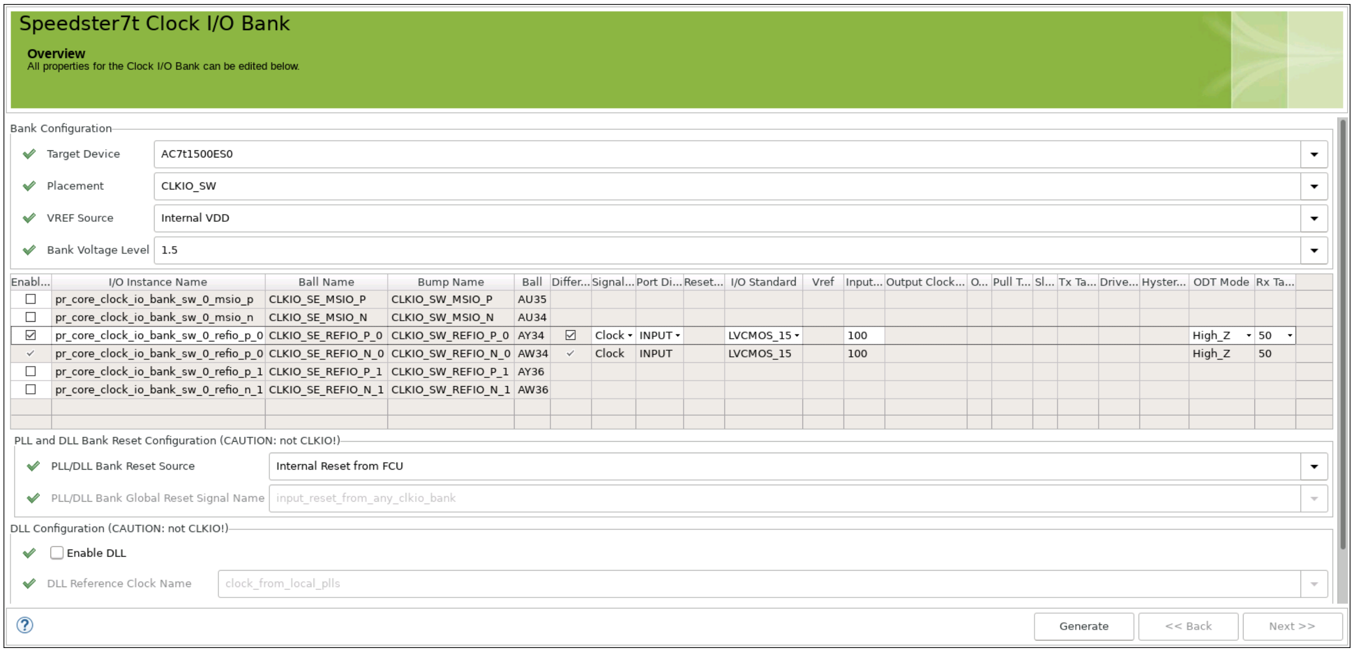


Figure 278: ***SW Clock I/O Bank IP Configuration Example***

3. In the **IP Libraries** view, under **IO Ring**, right-click the **PLL** and select **New IP Configuration**.
4. Configure the PLL for the PR core using the options shown in the following example:

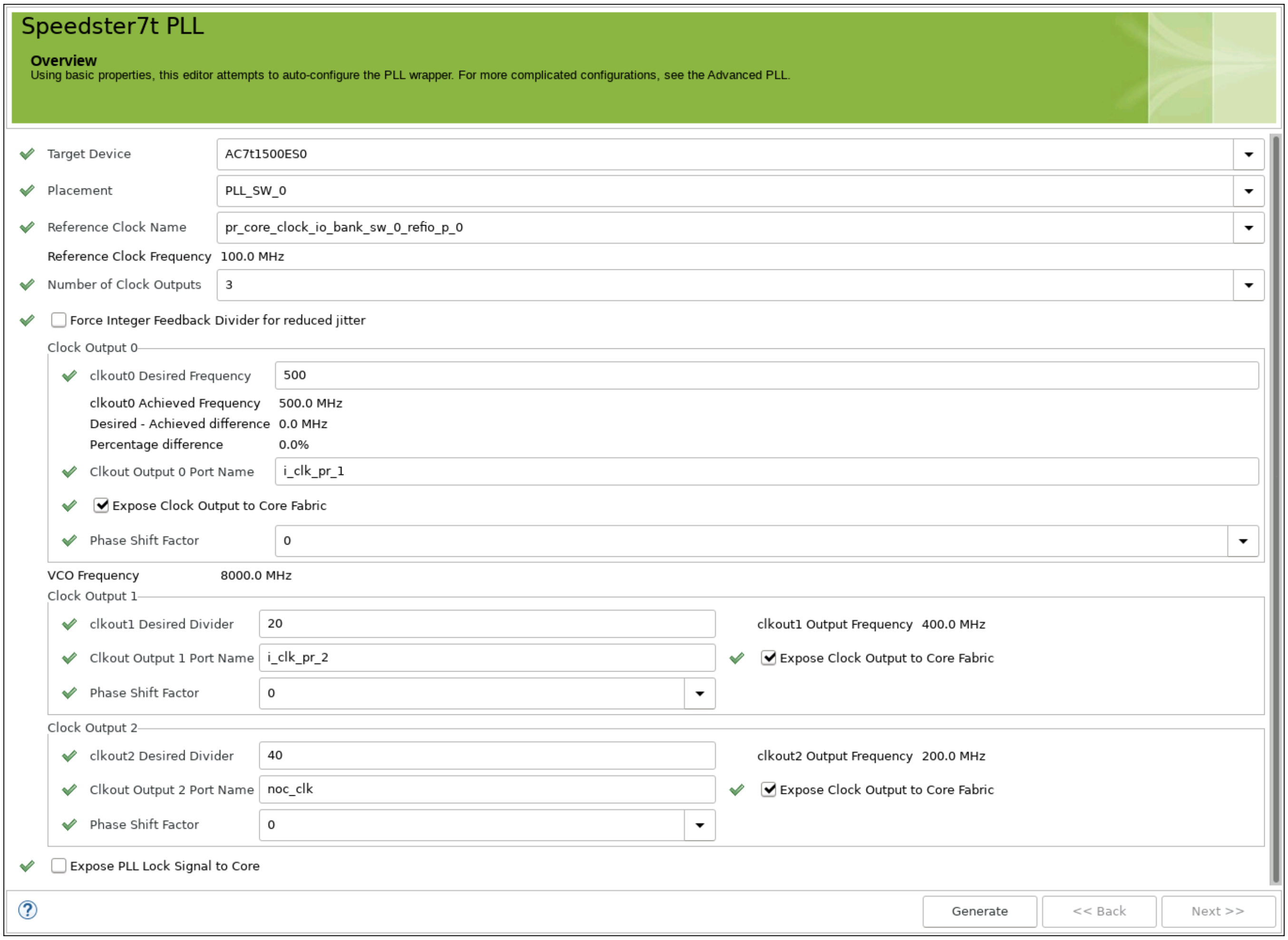


Figure 279: *SW PLL IP Configuration Example*

5. In the **IP Libraries** view, under **IO Ring**, right-click **NoC** and select **New IP Configuration**.
6. Configure the 2D NoC IP for the PR Core using the options shown in the following example:

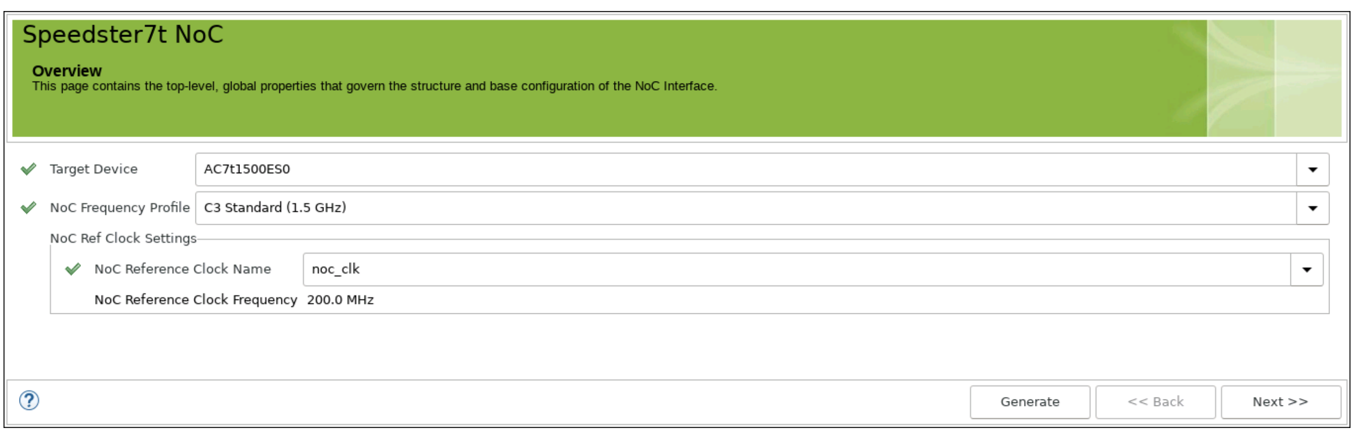


Figure 280: *2D NoC IP Configuration Example*

7. In the **IP Libraries** view, under **IO Ring**, right-click **Clock I/O Bank** and select **New IP Configuration**.
8. Configure the Clock I/O Bank for the top-level design using the options shown in the following example:

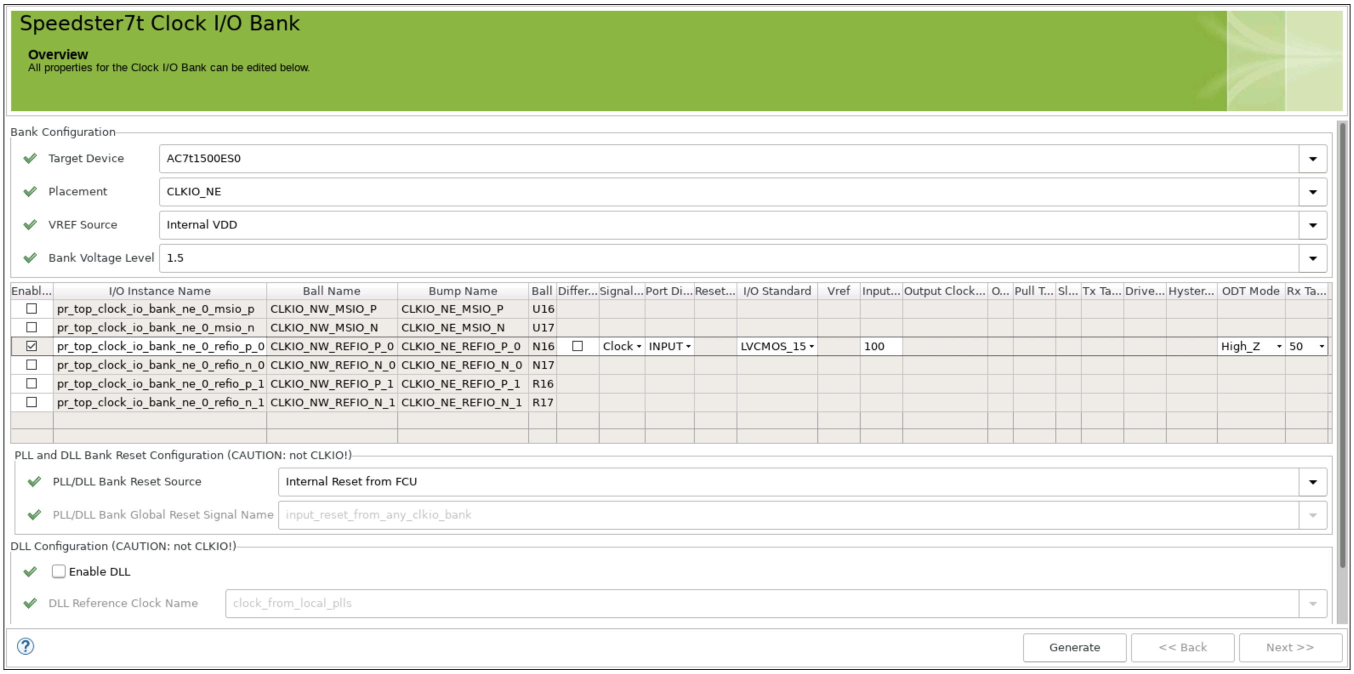


Figure 281: ***NE Clock I/O Bank IP Configuration Example***

9. In the **IP Libraries** view, under **IO Ring**, right-click **PLL** and select **New IP Configuration**.
10. Configure the PLL for the top-level design using the options shown in the following example:

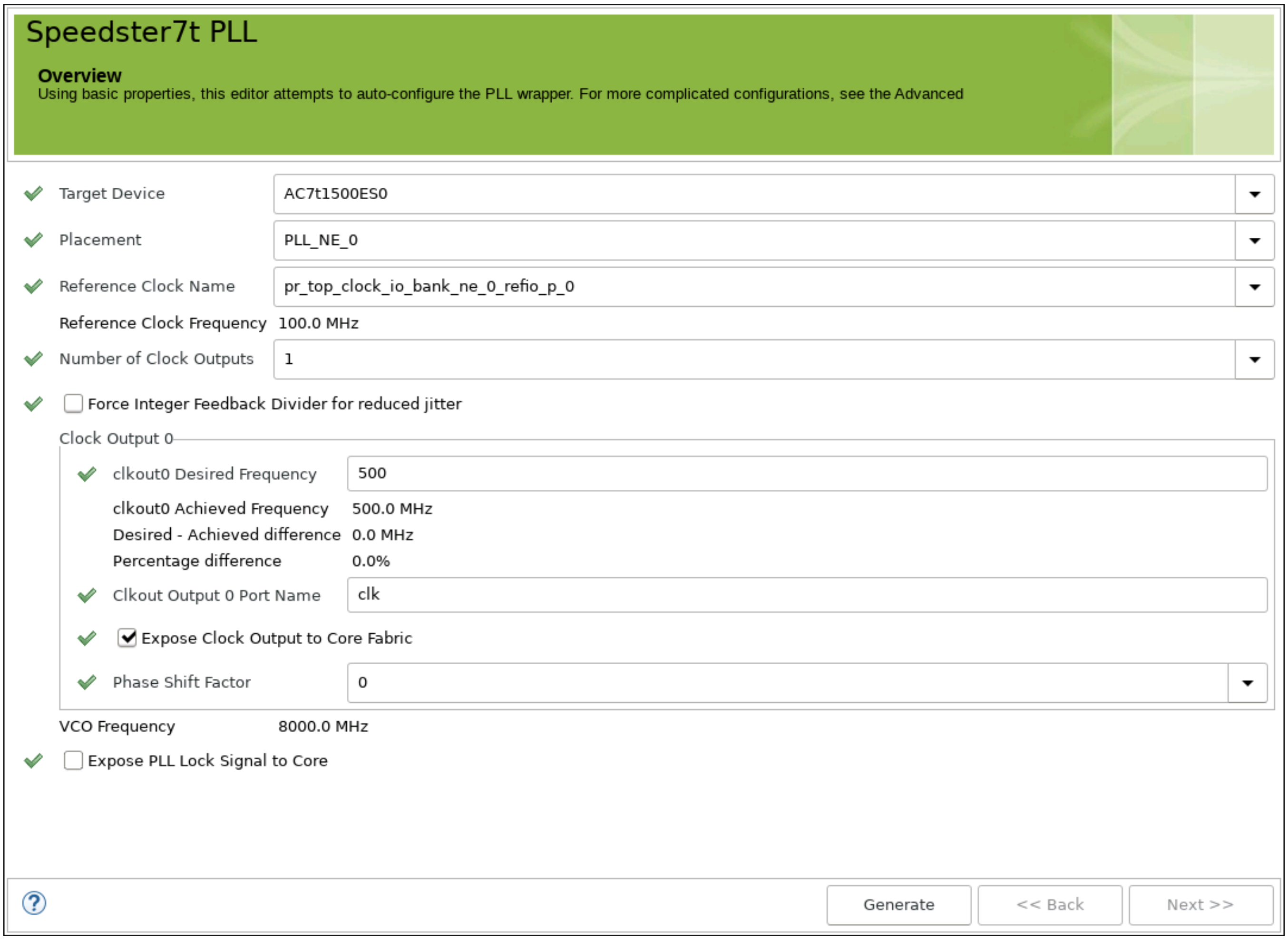


Figure 282: ***NE PLL IP Configuration Example***

11. In the **IP Libraries** view, under **IO Ring**, right-click **GPIO Bank** and select **New IP Configuration**.
12. Configure the GPIO Bank for the top-level design using the options shown in the following example:

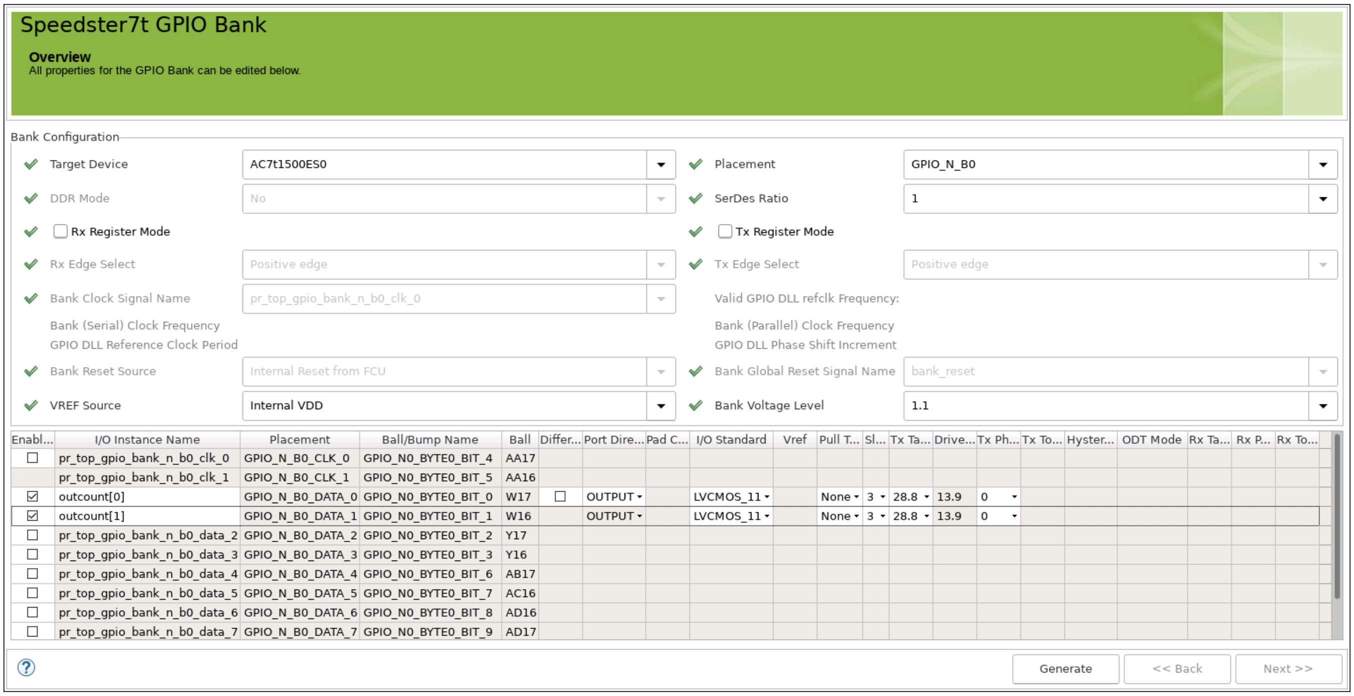


Figure 283: *GPIO Bank IP Configuration Example*

13. Click **Generate** to display the **Generate IO Ring Design Files** dialog and select **Add to active project**:

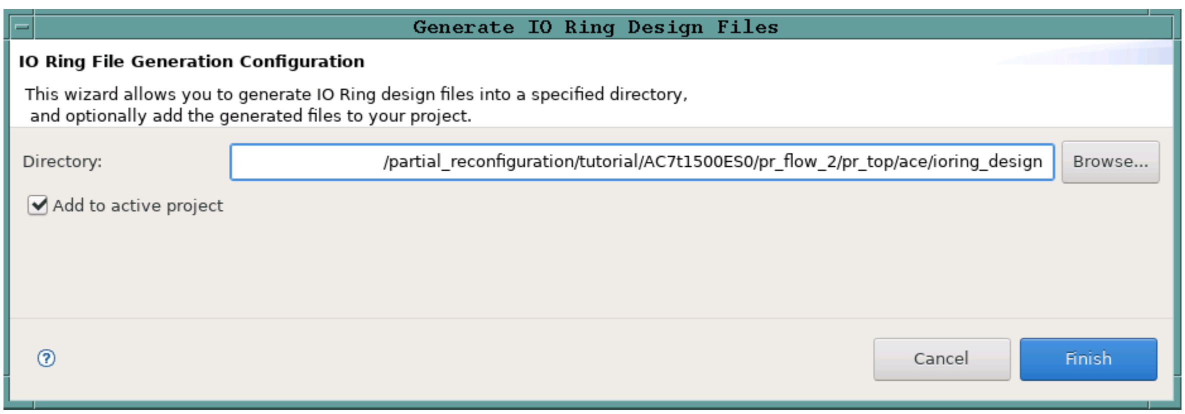


Figure 284: *Generate IO Ring Design Files Dialog Example*

14. Click **Finish** to generate the I/O ring design files.
15. When complete, the following files should be added to the ACE project:

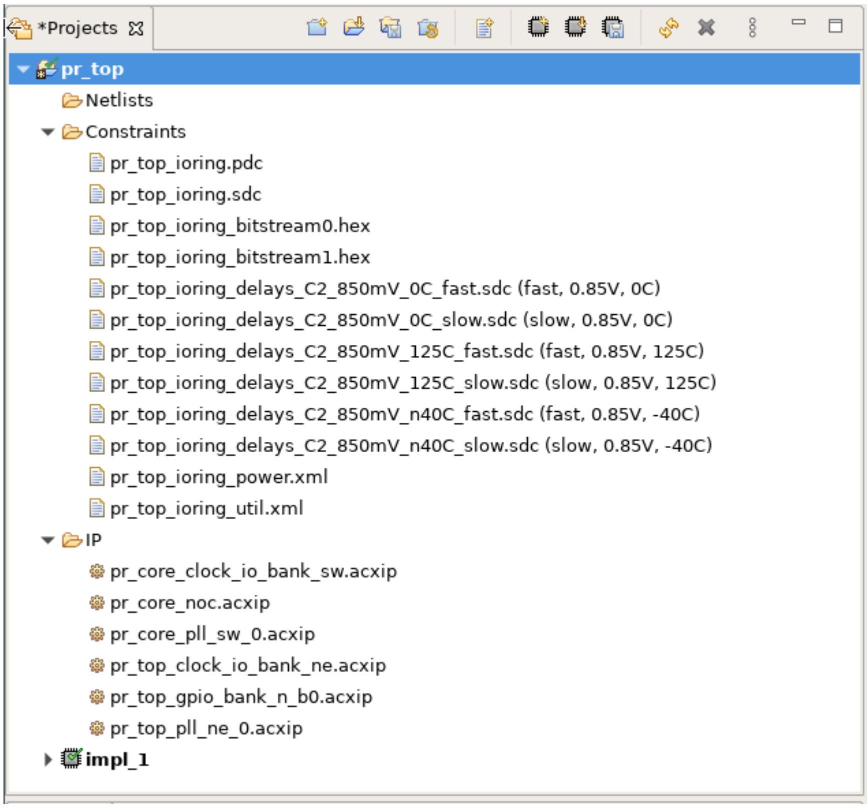


Figure 285: ***I/O Ring Design Files***

Synthesize Top-Level Design Using Synplify

1. Navigate to the following directory:
   `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src`
   Below this directory should be the following two directories:
   - `rtl`
   - `constraints`
2. Create a new Synplify project and add all RTL files from the `rtl` directory.
3. Add the `pr_top_timing.sdc` file from the `constraints` directory
4. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
5. Set **Part** to **AC7t1500ES0**.
6. Set **Package** to **F53**.
7. Set **Speed** to **C2**.
8. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_top**.
9. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_top**.
10. Set **Include Path Order** to `<ace_install_dir>/libraries`.
11. Add the `AC7t1500ES0_synplify.sv` file from the following directory:
    - `<ace_install_dir>/libraries/device_models`

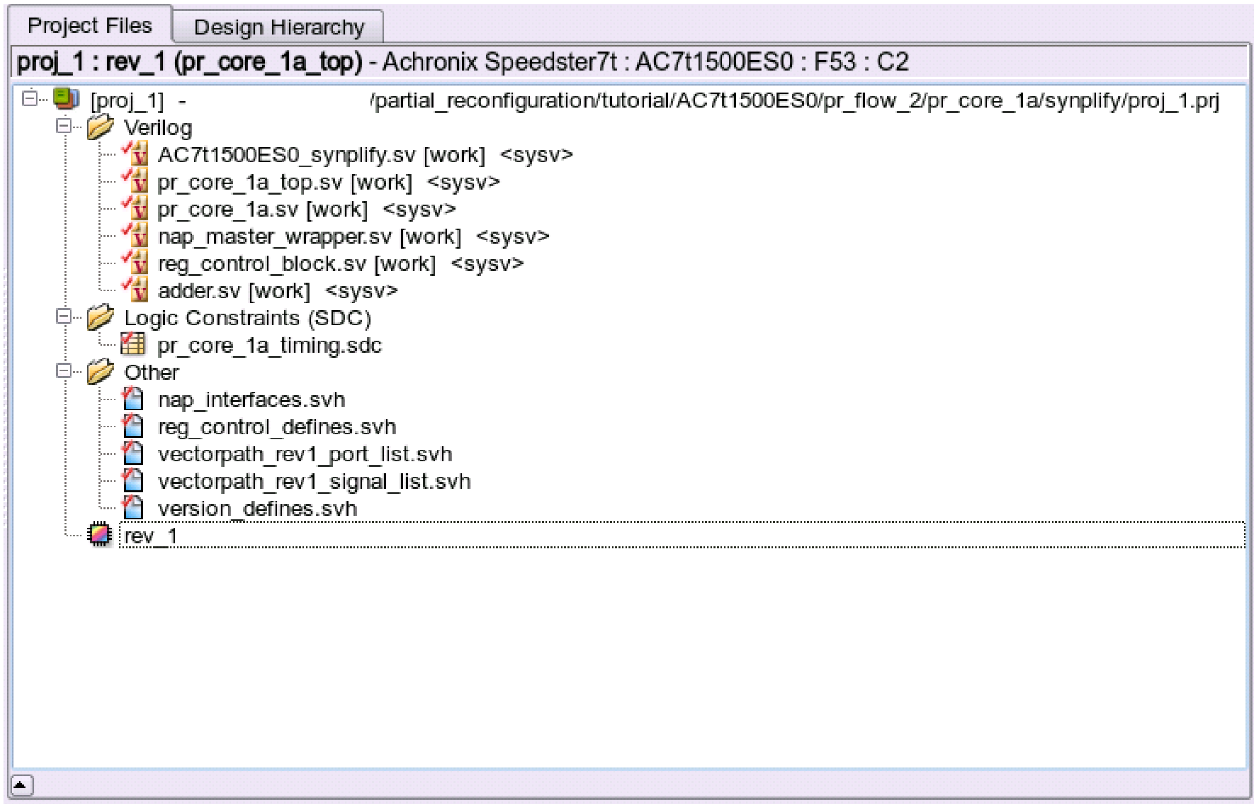


Figure 286: ***Synplify Project Files Example***

12. Run and compile the design using Synplify.
13. When the compile completes successfully, Synplify generates a gate-level netlist in the file: `<synplify_out_dir>/rev_1/pr_top.vm`.

Run Top-Level Design Through Run Prepare in ACE

1. In the **Projects** view, click the ( ) **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the Projects view and click the ( ) **Add Source Files to a Project** button.
4. Add the following files to the ACE project:
   - All the ioring files created in step 3
   - `<synplify_out_dir>/rev_1/pr_top.vm` (synthesized netlist created by Synplify in step 3)
5. In the Options view, select **Design Preparation** and make sure all options are correctly set.
6. Copy the option values in the following example:

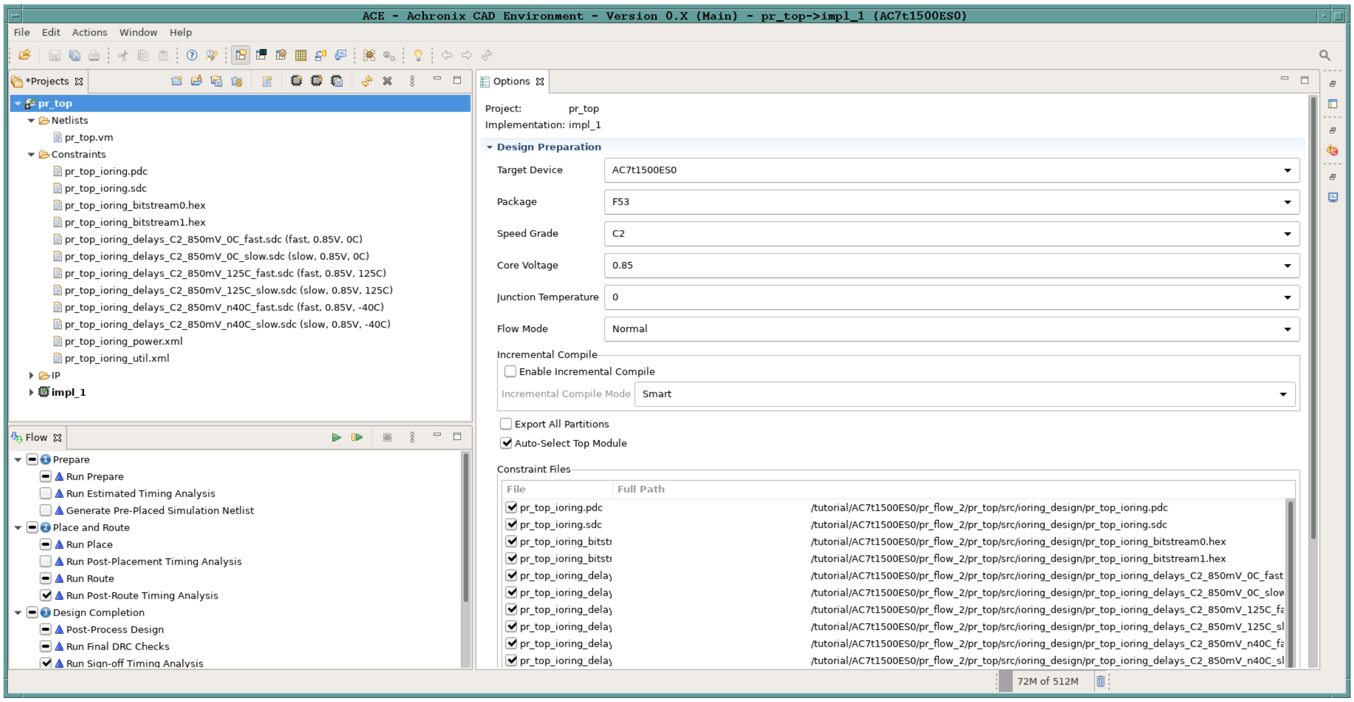


Figure 287: *Design Preparation Options Example*

7. In the Flow view, right-click the **Run Prepare** step and select **Run Selected Flow Step**.

Set Up Keep Out Regions and Placement Region Constraints for Static Logic

Keep out regions must be created in order to define "holes" in the core fabric that the top-level logic should not enter. These holes are later filled by partial reconfiguration bitstreams.

1. Ensure that the Run Prepare flow step has completed successfully.
2. In the **Floorplanner** view, click the () **Placement Region Tool** button.
3. Click and drag in the Floorplanner to draw a PR Zone for the PR Core.

> **Note**
>
> The bounding box of this keep out region must be the same as the one set for PR core 1A in section 1d.

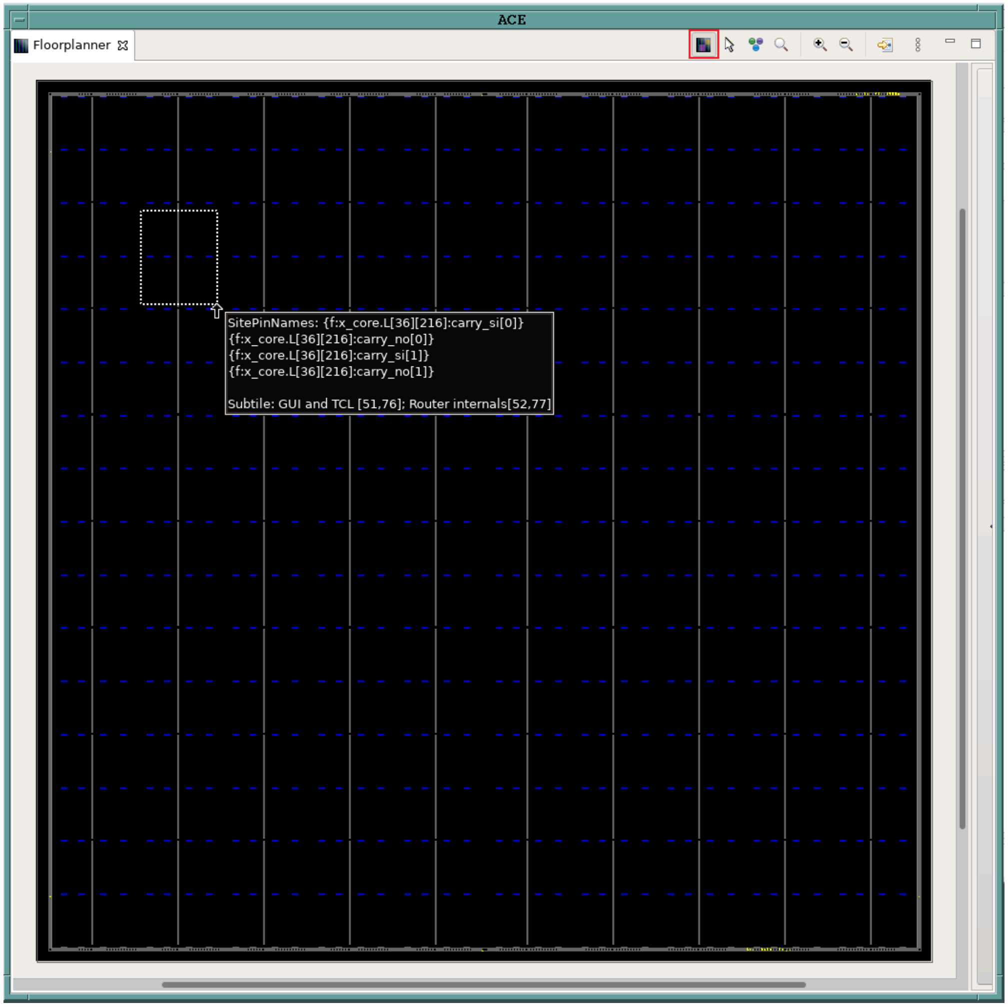


Figure 288: *Creating a Placement Region Zone Example*

4. After releasing the left mouse button, the **Create Placement Region** dialog appears.
5. Set **Region Alignment** to **Snap to Fabric Clusters**.
6. Set **Region Type** to **Keepout**.

> **Note**
>
> The bounding box of this region has to match that of the PR Zone for PR Core 1A created in step 1c.

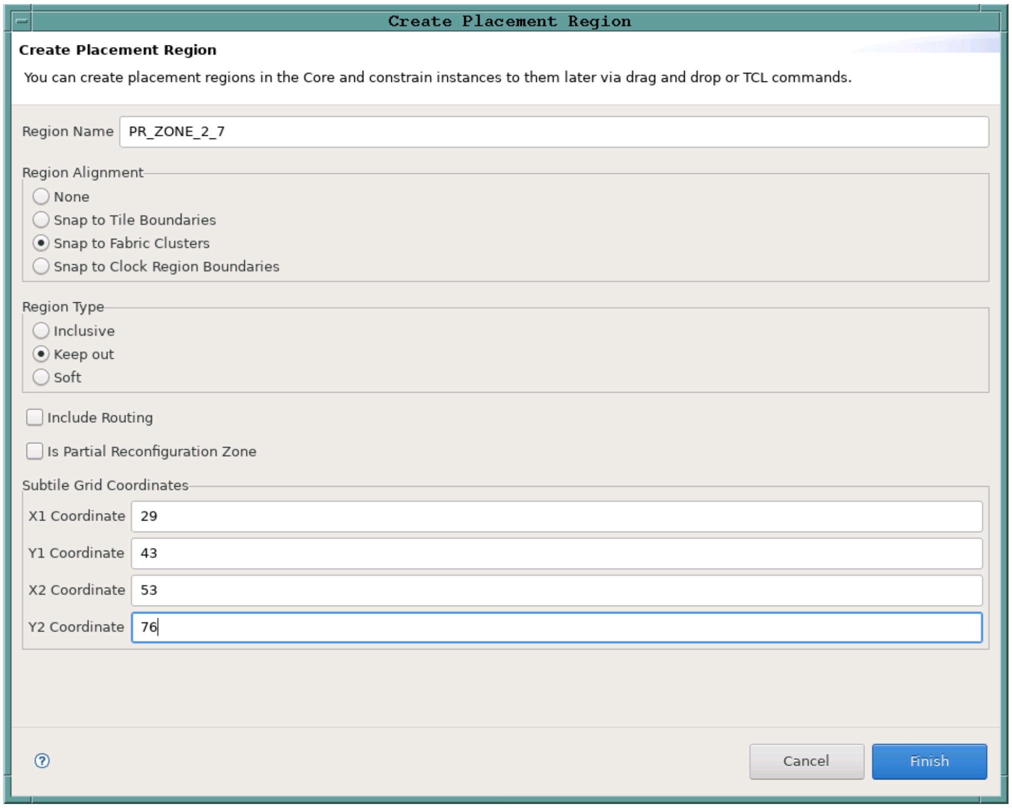


Figure 289: *Create Placement Region Dialog Example*

7. Repeat steps 2 through 6 to create a keep out region for the PR zone for PR core 2A.
8. Optionally, create a region for the top-level logic. The foorplanner view should resemble the following example after performing these steps:
   Red region – keep out region for PR core 1A.
   Light blue region – keep out region for PR core 2A.
   Yellow region – region for top-level logic.

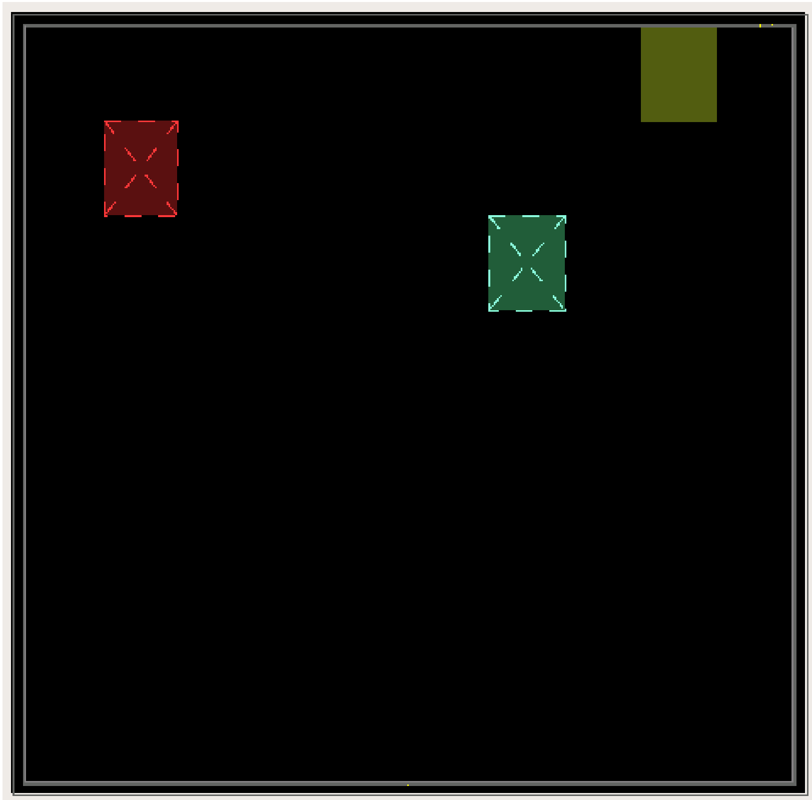


Figure 290: ***Floor Planner Regions Example***

Set Up Clock Pre-Routing Constraints

1. In the **Placement Regions** view, right-click the box below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes**:

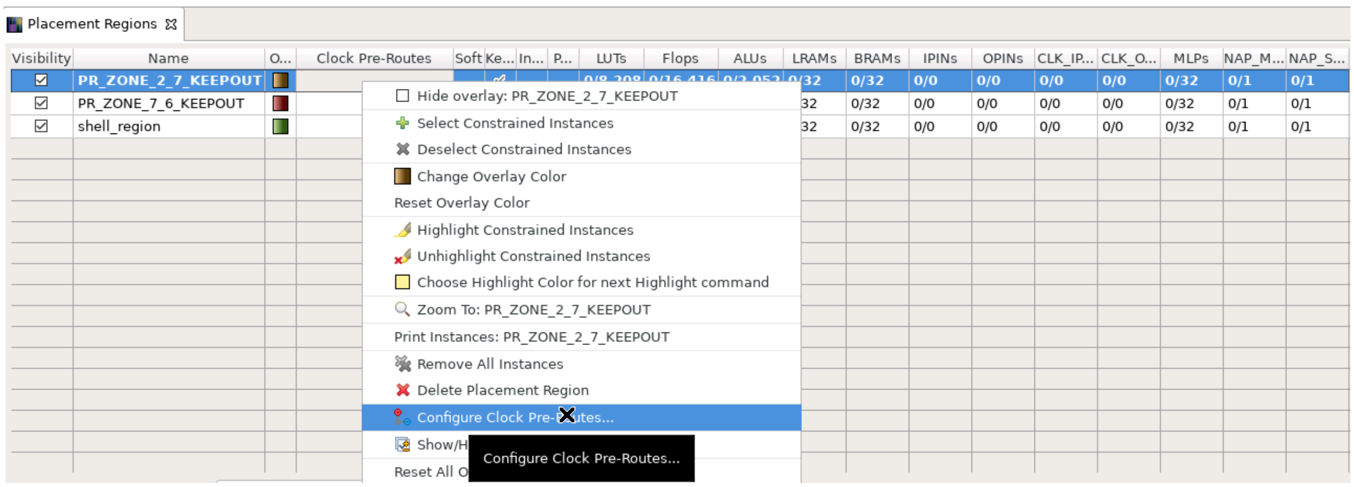


Figure 291: ***Configure Clock Pre-Routes Example***

2. The **Configure Clock Pre-Routes** dialog appears.
3. Ensure that the clock net is pre-routed into the same track specified in step 1d.

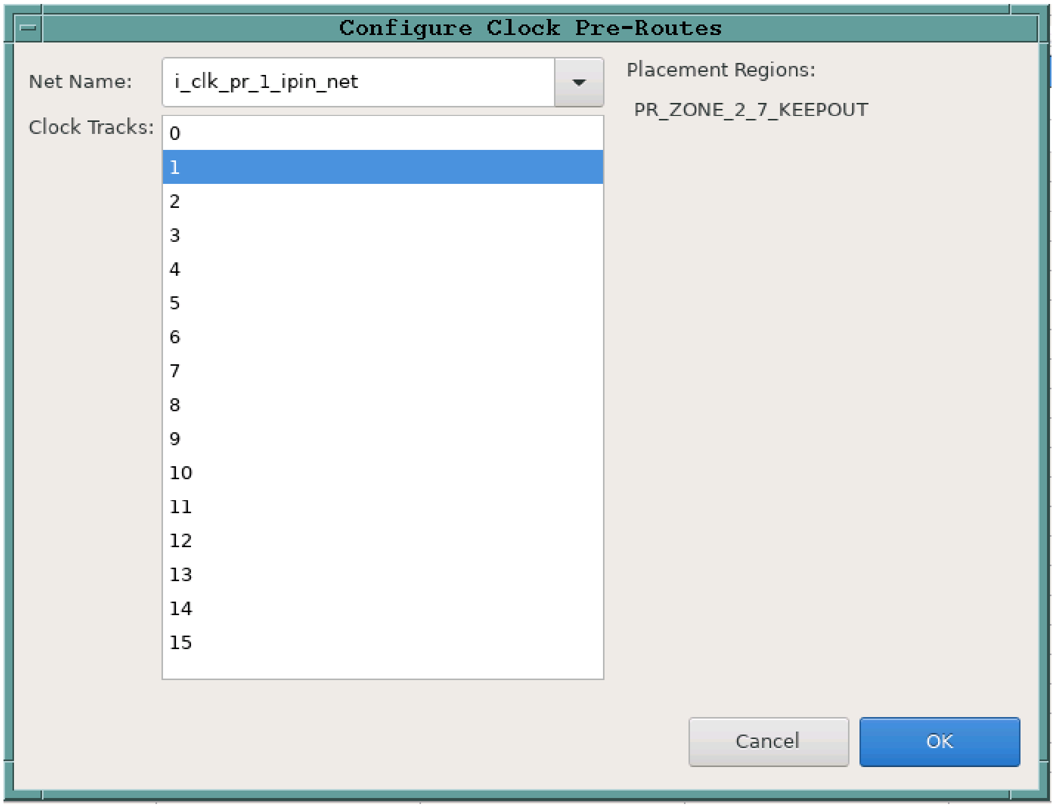


Figure 292: *Configure Clock Pre Routes Dialog Example*

4. Click **OK**. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -placement_regions { PR_ZONE_2_7 }
```

5. Repeat steps 1 and 2 above to route the clock net for PR core 2 to the PR core 2A keep out region.
6. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_2_ipin_net { 2 } -placement_regions { PR_ZONE_7_6 }
```

Run Place-and-Route in ACE for Top-Level Design

1. In the **Flow** view, right-click the **Place and Route** step and select **Run Selected Flow Step**.

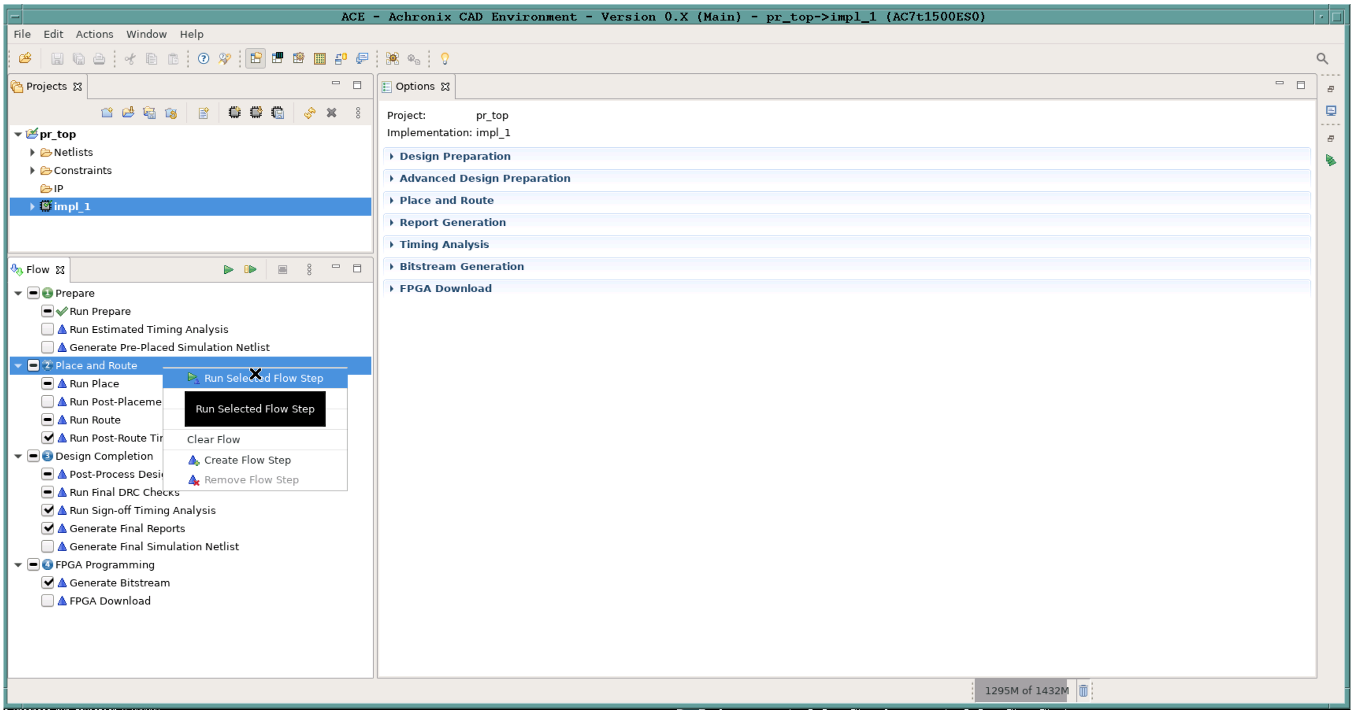


Figure 293: *Run Place and Route Flow Step Example*

Run Final Sign-Off Timing Analysis in ACE for Top-Level Design

1. Ensure that the Place and Route flow step completed successfully.
2. In the **Flow** view, right-click the **Design Completion** step and select **Run Selected Flow Step**.

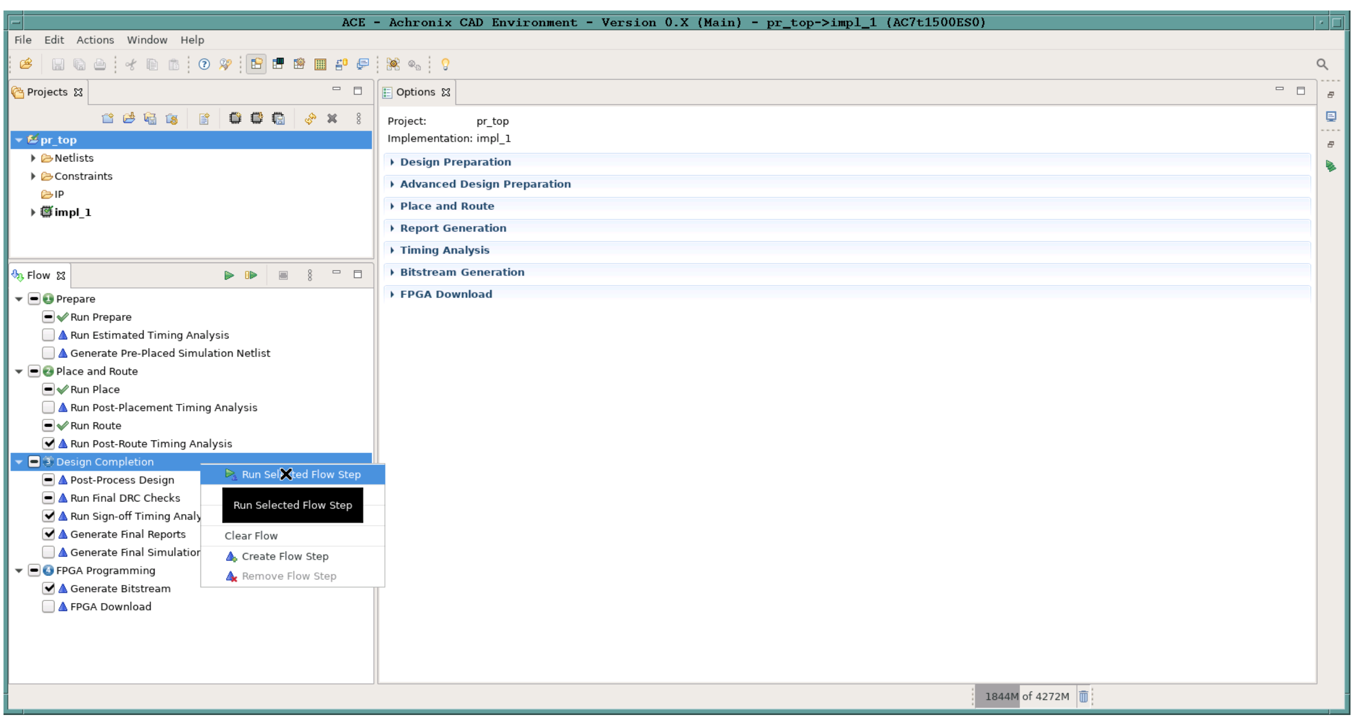


Figure 294: *Run Design Completion Flow Step Example*

Generate Base Bitstream for Top-Level Design

1. Ensure that the Design Completion flow step completed successfully.
2. In the Flow view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**.
3. Ensure that the **Enable Partial Reconfiguration** option is NOT selected in order to create a base bitstream.

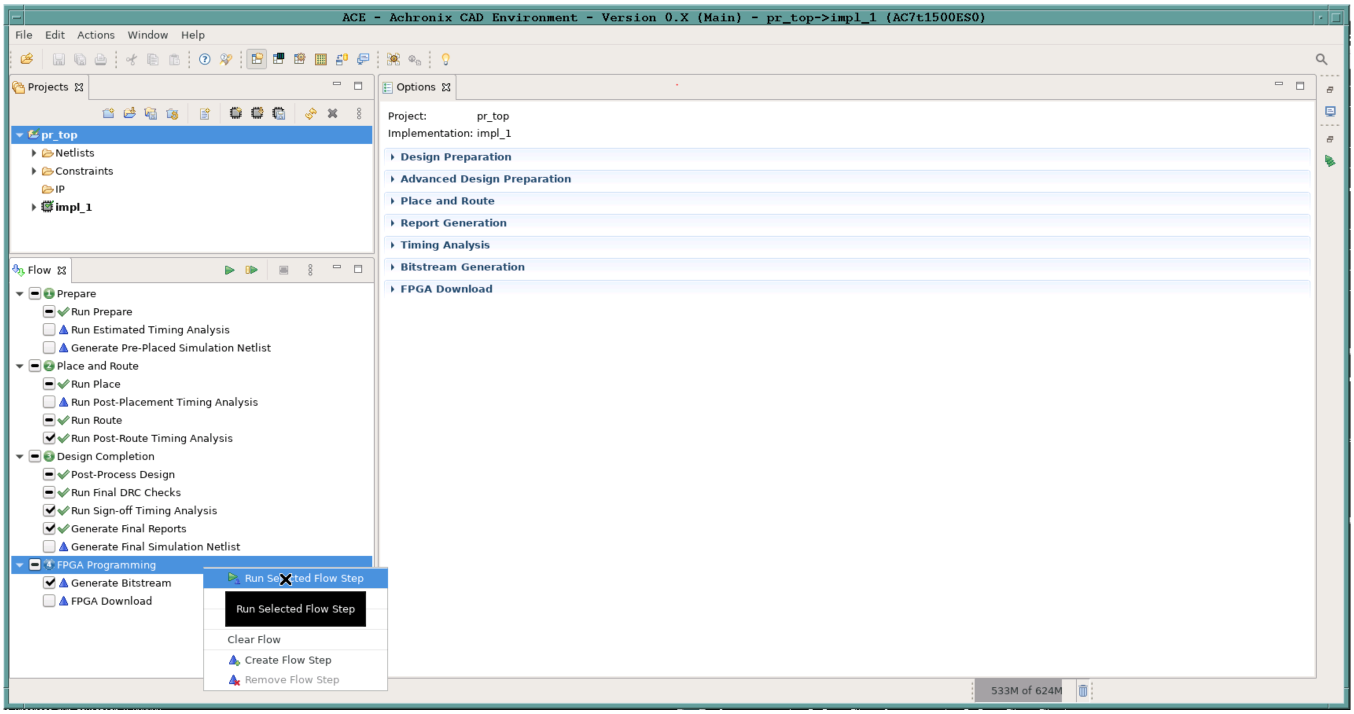


Figure 295: ***Run FPGA Programming Flow Step Example***

4. In the **Options** view, click the **Bitstream Generation** tab.
5. Scroll down to the **Partial Reconfiguration** section and ensure that the **Lock FCU After Programming** item is checked.

Figure 296: ***Lock FCU After Programming Option Example***

Save Placement Region and Clock Pre-Routing Constraints to PDC File

1. Repeat the steps in section 1i to save the region and clock pre-routing and placement region constraints to a `.pdc` file.

Bitstream Programming Sequence

Apply Power to Board

1. Apply power to the board and ensure cables are properly connected.

Program Top-Level Design Full Bitstream

1. Run the following commands.

```
set jtag_id [jtag::get_connected_devices]
jtag::open $jtag_id
jtag::configure_scan_chain $jtag_id AC7t1500ES0 0 0 0 -single_device
jtag::ac7t1500_initialize_fcu $jtag_id -reset
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_2/pr_top/ace/impl_1/output/pr_top.hex
```

Program PR Core 1A Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_2/pr_core_1a/ace/impl_1/output/pr_core_1a.hex
```

Run PR Core 1A Test Scripts

1. Run the following command.
   The script contains NAP reads from PR core 1A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/test_scripts
/read_pr_core_1a.tcl
```

Program PR Core 2A Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_2/pr_core_2a/ace/impl_1/output/pr_core_2a.hex
```

Run PR Core 2A Test Scripts

1. Run the following command.
   The script contains NAP reads from PR core 2A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/test_scripts
/read_pr_core_2a.tcl
```

Program PR Core 1B Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_2/pr_core_1b/ace/impl_1/output/pr_core_1b.hex
```

Run PR Core 1B Test Scripts

1. Run the following command.
   The script contains NAP reads from PR core 1B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/test_scripts
/read_pr_core_1b.tcl
```

Program PR Core 2B Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/partial_reconfiguration/tutorial
/AC7t1500ES0/pr_flow_2/pr_core_2a/ace/impl_1/output/pr_core_2b.hex
```

Run PR Core 2B Test Scripts

1. Run the following command.
   The script contains NAP reads from PR core 2B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/test_scripts
/read_pr_core_2b.tcl
```

Chapter - 4: Tcl Command Reference

The Tcl commands supported by ACE are broken into three subsets in this document:

- The SDC Commands (see page 509), timing constraints which are also supported by upstream tools like Synplify, These commands go in the SDC project constraints files.
- The Interactive Timing Commands (see page 528), commands which are used to interact with the ACE STA timer. The commands are not constraints and can be used interactively in the ACE Tcl console.
- The ACE Tcl Commands (see page 534), which are unique to ACE

SDC Commands

The following are the Tcl commands which are used to define timing constraints in both ACE and upstream tools like Synplify.

all_clocks

`all_clocks`

Returns a collection of all clocks in the design.

Description

The `all_clocks` command will return a list of all of the clocks that have already been defined using either `create_clock` or `create_generated_clock`. It is often used in SDC files as an argument to commands that need a list of all of the clocks.

Example

To set all of the clocks to have the same setup timing uncertainty value, enter:

```
cmd> set_clock_uncertainty -setup .05 [all_clocks]
```

Also See

`create_clock`

`create_generated_clock`

`get_clocks`

`set_clock_uncertainty`

all_inputs

`all_inputs`

Returns a collection of all input ports (ports marked "in" and "inout") at the top level of the design.

Description

The `all_inputs` command returns a list of all of the input ports in the design. It is sometimes used as a command line argument to other SDC commands when a list of all of the input ports are needed.

Example

To set all of the inputs to have the same minimum input delay from the same clock, enter:

```
cmd> set_input_delay -min .01 [all_inputs] -clock clk
```

Also See

`set_input_delay`

all_outputs

`all_outputs`

Returns a collection of all output ports (ports marked "out" and "inout") at the top level of the design.

Description

The `all_outputs` command returns a list of all of the output ports in the design. It is sometimes used as a command line argument to other SDC commands when a list of all of the output ports are needed.

Example

To set output delay constraint to all outputs with respect to the same clock, enter:

```
cmd> set_output_delay -min .01 [all_outputs] -clock clk
```

Also See

`set_output_delay`

create_clock

`create_clock [<clock>] [-period <string>] [-name <string>] [-waveform <list>]`

Define a clock

Argument	Optional	Description
`[<clock>]`	Y	nets, ports or pins (as 'inst/pin')
`[-period <string>]`	Y	clock period in ns (required)
`[-name <string>]`	Y	alternate name
`[-waveform <list>]`	Y	list of edges for clock rise and fall timings in the period

Description

The `create_clock` command is the main SDC constraint input to static timing analysis. In its simplest form it can define a clock and its associated period. This definition is used by the timer to start timing paths. The timing paths will take one of four possible paths:

1. Traverse from a create_clock statement, through the clock IPIN, through the clock tree to a source DFF/LRAM /BRAM clock pin, through the source device, through the whatever logic is between the source, to the capture logic or `set_output_delay` constraint.
2. Traverse from the create_clock statement, through a `set_input_delay` constraint defined on a given port, through the path from that port to the DFF/LRAM/BRAM data input pin. These paths are often referred to as I/O timing paths, and specifically, input timing paths.
    a. A `create_clock` statement can also be used strictly for IO timing, and not actually be placed and routed in the design. These are often refereed to as "virtual clocks".
3. Sometimes a `create_clock` statement will be assigned to an input port that will traverse to a data input pin of a DFF. If this is done, the arrival time of the rising and falling edges will be separated in time by the definitions of the first asserted edge and deasserted edge of the clock.

Example

To define a clock on an input clock port and assign it a period of 2 ns, enter:

```
cmd> create_clock -period 2 [get_ports clock_in[0]]
```

To define a clock with a non-default (50/50) duty cycle, the `create_clock -waveform` option can be used:

```
set clock_period 10
set clock_asserted_edge 0
set clock_deasserted_edge [expr $clock_period / 5]
create_clock -name my_clock_name -period $clock_period -waveform "$clock_asserted_edge
$clock_deasserted_edge" [get_ports my_clock_port_name]
```

To define a virtual clock you must use the `-name` option so that the clock name can be referenced by other SDC commands. Otherwise, the `-name` option is optional.

```
create_clock -name virtual_my_clock -period 10
```

The virtual_my_clock can then be used in IO timing constraints to define the arrival time of data at input ports based on a clock that will not have any design specific latency:

```
set_input_delay 1 -clock virtual_my_clock [get_ports i_user_data*]
```

Also See

`create_generated_clock`

`get_ports`

`report_clock_properties`

`report_clocks`

`report_checks`

`set_clock_latency`

`set_clock_groups`

`set_clock_uncertainty`

`set_input_delay`

`set_output_delay`

create_generated_clock

`create_generated_clock <clock> [-source <string>] [-divide_by <int>] [-multiply_by <int>] [-name <string>]`

Define a generated clock

Argument	Optional	Description
`<clock>`		nets or pins (as 'inst/pin')
`[-source <string>]`	Y	(required argument) master clock source pin
`[-divide_by <int>]`	Y	factor
`[-multiply_by <int>]`	Y	factor
`[-name <string>]`	Y	alternate name for the generated clock

Description

The `create_generated_clock` defines a clock which is applied to an output pin of an instance, internal to the design, or an output port of the design. This SDC command must follow a previously defined `create_clock` definition, of which the port used to define that `create_clock` statement would be used as the argument to the `-source` option. There must be a valid timing path between the source clock node and the generated clock node, so that latency between these two nodes can be calculated. A generated clock can have one of three characteristics of the source clock:

1. The same period as the source clock (`-divide_by` 1)
2. A period less than the source clock (`-divide_by` integer value greater than 1)
3. A period greater than the source clock (`-multiply_by` integer value greater than 1)

The generated clock will typically have a positive latency (delay) from the source clock as there is typically logic between the source clock and the generated clock. Therefore, the arrival time of a generated clock pin at a clock leaf node is calculated taking into account the latency from the source clock to the generated clock, plus the latency of the logic between the generated clock node and the generated clock leaf pin in the timing path. If frequency division is done (`-divide_by` or `-multiply_by`) than the deasserted edge and the second asserted edge of the generated clock's arrival times will be adjusted to the period calculated by the specified `-multiply_by` or `-divide_by` the respective values.

Example

To create a generated clock on an top level output port in a design, which is derived from a top level input port where the path is non-inverting and not divided, the following can be used:

```
create_generated_clock [get_ports o_user_clkout_001_003] -name out_clk -divide_by 1 -source [get_ports
i_user_clkin_001_003]
```

Also See

`create_clock`

`get_ports`

`report_clock_properties`

`report_clocks`

`report_checks`

`set_clock_groups`

`set_clock_latency`

`set_clock_uncertainty`

get_cells

`get_cells pattern`

Returns a collection of cells (instances) in the design. All cell names match the specified pattern. Wildcards may be used to select multiple cells at once.

Argument	Optional	Description
`pattern`		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_cells` command can be use in conjunction with other SDC commands when those commands need a list of cell instance names as input. It accepts the use of the "*" wildcard will return a list of all cell instance names that match.

Example

To get all cell instances with the string "reg" in their name, enter;

```
get_cells *reg*
```

The output of the `get_cells` command can be passed to other commands, such as `set_multicycle_path`:

```
set_multicycle_path -from [get_cells top.*my_module.module_reg*] -setup -end 2
```

Also See

`set_multicycle_path`

`set_false_path`

get_clocks

`get_clocks patterns [-nocase]`

Returns a collection of clocks in the design. All clock names in the collection match the specified pattern. Wildcards may be used to select multiple clocks at once.

Argument	Optional	Description
`patterns`		The required <patterns> option is used to filter returned node names (string patterns are matched using Tcl string matching)
`[-nocase]`	Y	The optional -nocase option specifies the matching of node names to the patterns should be case-insensitive

Description

The `get_clocks` command can be used after the `create_clock` command is used to define clocks. Additionally, if the `create_generated_clock` command is used, the `get_clocks` command will include them as well. The `get_clocks` command is used to get a sub-set of what would be returned by the `all_clocks` command. Typically, this command is used in conjunction with other SDC commands when a specific clock or specific group of clocks is needed as a command line argument.

Example

To get all of the clocks with the string "in" in their name, enter:

```
get_clocks *in*
```

To define clock to clock relationships, such as an asynchronous relationships between two clocks the following can be done:

```
set_clock_groups -asynchronous -group [get_clocks system_clock] -group [get_clocks test_clk]
```

Also See

`create_clock`

`create_generated_clock`

`report_clock_properties`

`report_clocks`

`set_clock_groups`

`set_clock_latency`

`set_clock_uncertainty`

get_fanout

`get_fanout [-flat] [-endpoints_only] [-only_cells] [-from <string>]`

Returns a collection of pins in the fanout of specified objects in the design.

Argument	Optional	Description
`[-flat]`	Y	Without this option, only pins at the same hierarchy level as the sinks are returned. With the option, pins in the fanout at any hierarchy level are returned.

Argument	Optional	Description
`[-endpoints_only]`	Y	Only return pins that are endpoints.
`[-only_cells]`	Y	Return the instances connected to the pins in the fanout.
`[-from <string>]`	Y	List of pins, ports, or nets to find the fanout of. For nets, the load pins on the nets are returned.

get_nets

`get_nets pattern`

Returns a collection of nets in the design. All net names in the collection match the specified pattern. Wildcards may be used to select multiple nets at once.

Argument	Optional	Description
`pattern`		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_nets` command can be use in conjunction with other SDC commands when those commands need a list of net names as input. It accepts the use of the "*" wildcard will return a list of all net names that match.

Example

To get all nets that have a name containing the string "data" and ending with "[0]", enter:

```
get_nets *data*[0]
```

To define a false path through a net the following command style can be used:

```
set_multicycle_path 2 -setup -through [get_nets *reset_sync_n] -to [get_clocks sys_clk]
```

Also See

`create_generated_clock`

`get_clocks`

`set_false_path`

`set_multicycle_path`

get_pins

`get_pins pattern`

Returns a collection of pins in the design. All pin names match the specified pattern. Wildcards may be used to select multiple pins at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_pins` command can be use in conjunction with other SDC commands when those commands need a list of cell instance pin names as input. It accepts the use of the "*" wildcard will return a list of all net names that match.

Example

In order to define a pin as an argument to another SDC command such as `create_generated_clock`, you can do the following, where the pin "clk_out" of the CLKDIV instance "sub-module top.first_sub_module.second_sub_module" is the location of the generated clock:

```
create_generated_clock -divide_by 2 [get_pins top.first_sub_module.second_sub_module/clk_out]
```

Likewise, the same method can be used for any SDC command that takes a pin argument:

```
set_multicycle_path -from [get_pins top.first_sub_module/*reg*/q] -to [get_pins top.second_sub_module/*re
g*/d] -setup 4
```

Also See

`create_generated_clock`

`set_false_path`

`set_multicycle_path`

get_ports

`get_ports pattern`

Returns a collection of ports (design inputs and outputs) in the design. All port names match the specified pattern. Wildcards may be used to select multiple ports at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_ports` command can be use in conjunction with other SDC commands when those commands need a list of top level port names as input. It accepts the use of the "*" wildcard will return a list of all ports names that match.

Example

To get all ports with the string "[0]" in their name, enter:

```
get_ports *[0]*
```

This command can also be used in an argument of other SDC commands that take a port and input. One of the most common is in the definition of a clock as it comes into the design:

```
create_clock -period 0.9 [get_ports {sys_clk}]
```

Also See

`create_clock`

`create_generated_clock`

`set_false_path`

`set_input_delay`

`set_multicycle_path`

`set_output_delay`

set_clock_groups

`set_clock_groups [-name <string>] [-group <list>] [-asynchronous]`

Define clock groups. With one -group, the clocks in that group have a false_path from/to all other clocks. With multiple -group options, the clocks in each group have a false_path from/to the clocks in the other groups. The groups have no meaning outside this command.

Argument	Optional	Description
`[-name <string>]`	Y	Name of clock group
`[-group <list>]`	Y	set of clocks
`[-asynchronous]`	Y	clocks are unrelated (default)

Description

The `set_clock_groups` command is defined in the SDC files after the `create_clock` and `create_generated_clock` statements have been defined. This command can be used to quickly define asynchronous relationships between clocks. This methodology replaced the older `set_false_path` based STA/SDC description methodology and is more efficient to write the SDC as well as enabling the timer to be more efficient.

Example

To assume a design has clocks system_clock and test_clk are asynchronous to all other clocks, enter:

```
set_clock_groups -asynchronous -group [get_clocks system_clock] -group [get_clocks test_clk]
```

This command specifies that A1 and B are unrelated to C. For instance, a path between A2 and C will not be timed. A path between A1 and A2, on the other hand, will be timed (unless there are other commands specifying a false path between them).

```
set_clock_groups -asynchronous -group [get_clocks {A B} -group [get_clocks C]
```

Also See

`create_clock`

`create_generated_clock`

`get_clocks`

`report_checks`

`set_false_path`

set_clock_latency

`set_clock_latency delay port_pin_list [-clock <string>] [-rise] [-fall] [-min] [-max] [-late] [-early] [-source]`

Set latency of clock network

Argument	Optional	Description
`delay`		delay_value
`port_pin_list`		port_pin_list (one or more ports)
`[-clock <string>]`	Y	clock list
`[-rise]`	Y	rise
`[-fall]`	Y	fall
`[-min]`	Y	min
`[-max]`	Y	max
`[-late]`	Y	late
`[-early]`	Y	early
`[-source]`	Y	source

Description

The `set_clock_latency` command is used to describe the arrival time of a clock at the top level port where it is defined using the `create_clock` command. The off-design latency of the clock will modify the timing of the IO logic. The impact of this can be seen in `report_checks` reports of IO timing where it will be reflected on both the reference clock path as well as the input data arrival times. It is common for there to be more than one `set_clock_latency` definitions for each clock in order to model the off design latency of the clock for both edges of the clock as well as the early and late arrival times of the clock. Care should be taken to ensure that early and late arrival times, which model the range of arrival times that can occur due to off design events such as crosstalk or varying paths to the clock input port, are not replicated (double counted) in the use of the `set_clock_uncertainty` command.

Example

In order to define off chip clock latency, which will impact clock to IO and clock to other clock timing, use the following command, for "late" arriving clock edges at the port where the acx_sc_i_user_clkin_000_001[0]_1 clock is defined:

```
set_clock_latency -source -late -rise 0.169006 [get_clocks acx_sc_i_user_clkin_000_001[0]_1]
```

Also See

`create_clock`

`get_clocks`

`report_clock_properties`

`report_clocks`

`report_checks`

`set_clock_uncertainty`

set_clock_uncertainty

`set_clock_uncertainty <uncertainty> [<objects>] [-from <string>] [-to <string>] [-setup] [-hold]`

Set uncertainty of clock network

Argument	Optional	Description
`<uncertainty>`		clock uncertainty in ns
`[<objects>]`	Y	one or more clocks, ports, or pins
`[-from <string>]`	Y	source clock
`[-to <string>]`	Y	destination clock
`[-setup]`	Y	uncertainty applies to setup check
`[-hold]`	Y	uncertainty applies to hold check

Description

> **Caution!**
>
> The `set_clock_uncertainty` command must be used with either the `<objects>` option or a pair of `-from` and `-to` options.

The `set_clock_uncertainty` command is generally used to model off design PLL jitter. This jitter is typically defined as "cycle to cycle" jitter, meaning that for one asserted edge to the next asserted edge there is some amount of +/-

"uncertainty" in the arrival time of the second asserted edge. This uncertainty is used to shorten (-) the clock insertion delay to a capture device in setup paths, and to increase the clock insertion delay to a capture device in hold timing paths. Since setup timing is typically measured from the first asserted edge to the next asserted edge, this PLL jitter based clock uncertainty is directly applicable. However, typically hold timing is done with respect to the same clock edge. Therefore, the `set_clock_uncertainty` command has both `-setup` and `-hold` options to enable the user to use different constraint values as the `-hold` value is not modeling PLL jitter, but instead can be used to add general timing guard band, which is typically referred to as modeling "known unknowns" as well as "unknown unknowns". The values of these constraints work in conjunction with the values defined in both the `create_clock` definitions, as well as the `set_clock_latency` values.

Care should be taken to ensure that uncertainty defined in those other constraint commands are not duplicated in the `set_clock_uncertainty` command. The effect that `set_clock_uncertainty` has on timing is global. All timing paths, both core and IO, will be impacted by this constraint and will be visible in the `report_checks` reports in the capture or "reference" clock timing path on the "clock uncertainty" line. For setup paths, the value will be subtracted from the clock arrival time, and for hold timing the value will be added to the clock arrival time.

Example

To model PLL cycle to cycle jitter specification of 0.02nS for all of the clocks, the following command can be used:

```
set_clock_uncertainty -setup .02 [all_clocks]
```

To define extra hold timing guard band the following command can be used:

```
set_clock_uncertainty -hold .005 [all_clocks]
```

Also See

`all_clocks`

`create_clock`

`create_generated_clock`

`report_clock_properties`

`report_clocks`

`report_checks`

`set_clock_latency`

set_data_check

`set_data_check value [-clock <string>] [-setup] [-hold] [-from <string>] [-to <string>]`

Set data-to-data check values of setup and hold

Argument	Optional	Description
`value`		check value
`[-clock <string>]`	Y	clock
`[-setup]`	Y	setup

Argument	Optional	Description
`[-hold]`	Y	hold
`[-from <string>]`	Y	from_list (one or more clocks)
`[-to <string>]`	Y	to_list (one or more clocks)

Description

The `set_data_check` command can be used to add timing constraint between two data signals arriving to different pins /ports. This added timing constraint is analogous to the standard setup/hold timing constraints between a clock and data modeled for a DFF/LRAM/BRAM, but in this case the `-from` related pins is defined as the reference or clock pin, and the `-to` related pin is the data pin. The command supports unique `-setup` and `-hold` values, but if neither `-setup` nor `-hold` options are used, the values are applied only to setup. Often this command is used to define "data skew" which is typically defined as a +/- delta between data bus arrival times. Therefore, both the `-setup` and `-hold` options must be used, in different command instantiations, to define one data bit in a bus as the reference to N number of other data bits. Both the `-from` pin and the `-to` pin must be singular. For a bus that is 16 bits wide, there needs to be 15 constraints.

The `-from` and `-to` nodes defined in these commands must have existing valid timing paths to them for the `set_data_check` command to function. Therefore, if these constraints are applied to output ports, there must also be a `set_output_delay` constraint applied to them. Additionally, if both `-setup` and `-hold` data checks are to be performed, there must be both `-min` and `-max set_output_delay` constraints defined.

Example

In order to constrain the delay between two or more data pins, such as a "data skew" constraint, the following command can be used. This will define both a setup and hold timing relationship between the reference_pin_name and all of the the other_pin_names.

```
set_data_check -from [get_pins top.sub-module.instance1/reference_pin_name] -to [get_pins top.sub-module.
instance1/another_pin_name] .1
```

If more than one clock can drive a signal to the `-from` related pin, than the command can be made more specific by using the `-clock` options

```
set_data_check -from [get_pins top.sub-module.instance1/pin_name] -to [get_pins top.sub-module.instance1
/another_pin_name] .1 -clock [get_clocks my_clock]
```

To model a data skew constraint, a Tcl loop such as this can be used:

```
set first_port 0
set plus_minus_constraint 0.05
foreach port_name [get_ports dout_gpio[*]] {
    if { $first_port == 0 } {incr first_port;set reference_port $port_name;continue}
    set_data_check -from $reference_port -to $port_name $plus_minus_constraint -hold
    set_data_check -from $reference_port -to $port_name $plus_minus_constraint -setup
}
```

Also See

`get_pins`

`get_clocks`

`report_checks`

`set_output_delay`

set_disable_timing

`set_disable_timing <objects> [-from <string>] [-to <string>]`

Disable timing arcs in a circuit

Argument	Optional	Description
`<objects>`		one or more instances, ports, or pins
`[-from <string>]`	Y	input pin name of instance <object>
`[-to <string>]`	Y	output pin name of instance <object>

Description

The `set_disable_timing` command is typically used to disable an existing timing arc. This is somewhat analogous to `set_false_path`, but with a much more limited scope. The `set_disable_timing` command requires a `-from` and a `-to` option to be used together, to bound the scope of its effect. Often, this command is used to break timing loops from an input pin to an output pin of the same cell instance. The pin names used in the `-from` and `-to` options must be just the pin name, as found in the cell library.

Example

In order to break all of the timing arcs for an instance (top.sub-module.instance1), the following command can be used:

```
set_disable_timing [get_cells top.sub-module.instance1]
```

To break a given timing arc between two pins of a given cell instance, the following can be done:

```
set_disable_timing -from input_pin_name -to input_pin_name [get_cells top.sub-module.instance1]
```

Also See

`get_cells`

`set_false_path`

set_false_path

`set_false_path [-from <list>] [-to <list>] [-through <list>]`

Define a false path exception (this declares that the clocks are unrelated)

Argument	Optional	Description
`[-from <list>]`	Y	from_list (one or more clocks)

Argument	Optional	Description
`[-to <list>]`	Y	to_list (one or more clocks)
`[-through <list>]`	Y	through_list (one or more clocks)

Description

The `set_false_path` command is used to create "timing exception" to the general STA paradigm that all timing paths are analyzed as a one cycle setup and zero cycle hold timing path. Another timing exception syntax is `set_multicycle_path` and the user must be sure if a path is truly never used, or if it is a multicycle path. Timing paths that are analyzed by STA, but found to not be valid for whatever reason can be removed from the analysis by using the `set_false_path` statement. This command has several options, and can define very wide ranging paths so care should always be taken to limit the scope of the these commands in order to ensure that only the paths known to be false are affected by these commands. To enable users to focus these statements on a finite number of paths there is syntax to define the path start point (`-from`), path intermediate points (`-through`) as well as path end points (`-to`). It is advisable to be as explicit in the timing path definition as possible to ensure that real or valid timing paths are not being suppressed. Often the process of defining timing exceptions is like "peeling an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next-worst path becomes the new worst case path. Therefore, the most effective timing exceptions are typically constructed after all of the timing paths have been validated and all of the resulting exceptions combined to minimize the number of exceptions.

The definition of the path can be very narrow or very broad depending on how it is constructed. Each statement can contain only one `-from` and one `-to` statement, but each of them can reference many "from" nodes and many "to" nodes. If there are more than one node for these, all combinations apply. All "from" nodes are applied to all "to" nodes. Additionally, the `-through` option can have multiple nodes defined in it. Each of the `-through` nodes apply independently so if a path goes through any of the matching nodes it applies. However, multiple `-through` statements can be used in series with each other. They are order dependent, so '`-through a`' and '`-through b`' implies that the path must first go through "a" and then go through "b". If the `-through` command has multiple nodes in it, than that one statement is define as an 'OR'; '`-through a -through "b c" -through d`' implies that the path must go through "a" and then go through either "b" or "c", and then go through "d".

Example

In order to remove a timing path between an "instance/clock_pin_name" to an "instance/input_pin" from being analyzed by the timer, the following command can be used:

```
set_false_path -from [get_pins top.module1.reg_instance_name/clock_pin_name] -through [get_nets
some_applicable_net] -to [get_pins top.module2.instance_name/input_pin_name]
```

In order to remove all timing from a given timing node such as an input port, the following can be done:

```
set_false_path -from [get_ports my_port_which_I_do_not_want_to_time]
```

A false path can also be define `-through` a net, as well as instances and pins:

```
set_false_path -through [get_nets my_net_of_interest_name]
```

Also See

`get_pins`

`get_ports`

`get_nets`

`set_multicycle_path`

set_input_delay

`set_input_delay delay port_pin_list [-clock <string>] [-rise] [-fall] [-max] [-min] [-add_delay] [-clock_fall]`

Specify an input delay constraint or clock

Argument	Optional	Description
`delay`		delay_value
`port_pin_list`		port_pin_list (one or more ports)
`[-clock <string>]`	Y	clock_name
`[-rise]`	Y	rise
`[-fall]`	Y	fall
`[-max]`	Y	max
`[-min]`	Y	min
`[-add_delay]`	Y	add delay
`[-clock_fall]`	Y	delay with reference to falling edge of clock

Description

The `set_input_delay`, as well as the `set_output_delay` command, is fundamental to validating the correctness of a design's timing. It is defined after the clocks are define and it references the related clock using the `-clock` option. Typically, there will be four (4) definitions of `set_input_delay` for each data/clock combination, at each timing corner. The arrival time of the data at its design input port is relative to this clock's arrival time. Therefore, the `set_clock_latency` constraint will impact the arrival time of data related to that clock. The value of the `set_input_delay` constraint is used in the timing path which receives the data signal. This value can be seen in a `report_checks` report in the data path section under "input external delay".

Example

In order to constrain a design's input port, an arrival time for a signal at the input port, relative to the asserted edge of a specified clock, for both min and max data path timing, as well as having different values for rise and fall edges, can be defined using this command:

```
set_input_delay .1 -rise -max -clock [get_clocks my_clock_name] [get_ports my_input_port_name]
set_input_delay .13 -fall -max -clock [get_clocks my_clock_name] [get_ports my_input_port_name]
set_input_delay .05 -rise -min -clock [get_clocks my_clock_name] [get_ports my_input_port_name]
set_input_delay .055 -fall -min -clock [get_clocks my_clock_name] [get_ports my_input_port_name]
```

Also See

`create_clock`

`get_clocks`

`get_ports`

`report_checks`

`set_clock_latency`

`set_output_delay`

set_input_transition

`set_input_transition slew port_pin_list [-clock <string>]`

Specify an input slew/transition constraint

Argument	Optional	Description
`slew`		slew_value
`port_pin_list`		port_pin_list (one or more ports)
`[-clock <string>]`	Y	clock_name

set_load

`set_load load port_pin_list`

Specify an output load/capacitance constraint

Argument	Optional	Description
`load`		cap_value
`port_pin_list`		port_pin_list (one or more ports)

set_max_delay

`set_max_delay delay [-from <list>] [-to <list>] [-through <list>]`

Set a maximum delay for a path

Argument	Optional	Description
`delay`		delay
`[-from <list>]`	Y	from_list (one or more clocks)
`[-to <list>]`	Y	to_list (one or more clocks)
`[-through <list>]`	Y	through_list (one or more clocks)

set_min_delay

`set_min_delay delay [-from <list>] [-to <list>] [-through <list>]`

Set a minimum delay for a path

Argument	Optional	Description
`delay`		delay
`[-from <list>]`	Y	from_list (one or more clocks)
`[-to <list>]`	Y	to_list (one or more clocks)
`[-through <list>]`	Y	through_list (one or more clocks)

set_multicycle_path

`set_multicycle_path multiplier [-setup] [-hold] [-start] [-end] [-from <list>] [-to <list>] [-through <list>]`

Define multicycle path

Argument	Optional	Description
`multiplier`		integer path multiplier
`[-setup]`	Y	Use the specified path multiplier for setup/max delay calculations. This is the default behavior
`[-hold]`	Y	Use the specified path multiplier for hold/min delay calculations
`[-start]`	Y	Use the start/source clock period in the calculations. This is the default behavior for hold checks
`[-end]`	Y	Use the end/target clock period in the calculations. This is the default behavior for setup checks
`[-from <list>]`	Y	list of path startpoints containing clock, primary input/inout port, sequential-cell instance, clock pin of sequential-cell instance, or pin with input delay constraint
`[-to <list>]`	Y	list of path endpoints containing clock, primary output/inout port, sequential-cell instance, data pin of sequential-cell instance, or pin with output delay constraint
`[-through <list>]`	Y	list of pins, ports, nets that the path must pass through

Description

The `set_multicycle_path` command is used to create "timing exception" to the general STA paradigm that all timing paths are analyzed as a one cycle setup and zero cycle hold timing path. Another timing exception syntax is `set_false_path` and the user must be sure if a path is truly used but in more than one cycle, or never used. Timing paths that are analyzed by STA, but found to not be only valid on more than one cycle, for whatever reason, can have it's analysis adjusted by using the `set_multicycle_path` statement. This command has several options, and can define very wide ranging paths so care should always be taken to limit the scope of the these commands in order to ensure that only the paths known to be false are effected by these commands. To enable users to focus these statements on a finite number of paths there is syntax to define the path start point (`-from`), path intermediate points (`-through`) as well as path end points (`-to`). It is advisable to be as explicit in the timing path definition as possible to ensure that real or valid timing paths are not being suppressed. Often the process of defining timing exceptions is like "peeling an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next-worst path becomes the new worst case path. Therefore, the most effective timing exceptions are typically constructed after all of the timing paths have been validated and all of the resulting exceptions combined to minimize the number of exceptions.

The definition of the path can be very narrow or very broad depending on how it is constructed. Each statement can contain only one `-from` and one `-to` statement, but each of them can reference many "from" nodes and many "to" nodes. If there are more than one node for these, all combinations apply. All "from" nodes are applied to all "to" nodes. Additionally, the `-through` option can have multiple nodes defined in it. Each of the `-through` nodes apply independently so if a path goes through any of the matching nodes it applies. However, multiple `-through` statement can be used in series with each other. They are order dependent, so '`-through a`' and '`-through b`' implies that the path must first go through "a" and then go through "b". If the `-through` command has multiple nodes in it, than that one statement is define as an 'OR'; '`-through a -through "b c" -through d`' implies that the path must go through "a" and then go through either "b" or "c", and then go through "d".

Example

To change the default STA one cycle timing paradigm for all paths between two DFFs, to being a two cycle path, the following can be done:

```
set_multicycle_path 2 -from [get_pins top.sub_module_name.register_name/ck] -to [get_pins top.
some_module_name.some_register_name/q] -setup
```

It is important to understand, that the changing of the default one cycle path for setup, usually requires a modification from the default zero (0) cycle hold timing constraint (but not always). In this case, the hold timing is changed to be a one (1) cycle hold check:

```
set_multicycle_path 1 -from [get_pins top.sub_module_name.register_name/ck] -to [get_pins top.
some_module_name.some_register_name/q] -hold
```

Also See

`get_pins`

set_output_delay

`set_output_delay delay port_pin_list [-clock <string>] [-rise] [-fall] [-max] [-min] [-add_delay] [-clock_fall]`

Specify an output delay constraint or clock

Argument	Optional	Description
`delay`		delay_value
`port_pin_list`		port_pin_list (one or more ports)
`[-clock <string>]`	Y	clock_name
`[-rise]`	Y	rise

Argument	Optional	Description
[-fall]	Y	fall
[-max]	Y	max
[-min]	Y	min
[-add_delay]	Y	add delay
[-clock_fall]	Y	delay with reference to falling edge of clock

Description

The `set_output_delay`, as well as the `set_input_delay` command, is fundamental to validating the correctness of a design's timing. It is defined after the clocks are define and it references the related clock using the `-clock` option. Typically, there will be four (4) definitions of `set_output_delay` for each data/clock combination, at each timing corner. The required time of the data at its design output port is relative to this clock's arrival time. Therefore, the `set_clock_latency` constraint will impact the required time of data related to that clock. The value of the `set_output_delay` constraint is used in the timing path which drives the output data signal. This value can be seen in a `report_checks` report in the data path section under "output external delay".

Example

In order to constrain a design's output port, a required time for a signal at the output port, relative to the asserted edge of a specified clock, can be defined using this command:

```
set_output_delay .1 -rise -max -clock [get_clocks my_clock_name] [get_ports my_output_port_name]
set_output_delay .13 -fall -max -clock [get_clocks my_clock_name] [get_ports my_output_port_name]
set_output_delay .05 -rise -min -clock [get_clocks my_clock_name] [get_ports my_output_port_name]
set_output_delay .055 -fall -min -clock [get_clocks my_clock_name] [get_ports my_output_port_name]
```

Also See

`get_ports`

`get_clocks`

`report_checks`

`set_clock_latency`

`set_output_delay`

Interactive Timing Commands

These commands are used to query the ACE Static Timing Analyzer (STA) interactively, from the ACE command prompt. To use these commands, interactive timer mode must be enabled by calling prepare_sta (see page 530). To exit interactive timer mode, call reset_sta (see page 533). While in interactive timer mode, regular ACE commands remain available, but the placement and routing of the design should not be changed.

check_setup

The check_setup command performs sanity checks on the design. Individual checks can be performed with the keywords. If no check keywords are specified all checks are performed.

```
Command Syntax

check_setup [-verbose] [-unconstrained_endpoints] [-multiple_clock] [-no_clock] [-no_input_delay] [-
no_output_delay] [-loops] [-generated_clocks]
```

Table 153: *Command-line Options for check_setup*

Argument	Optional	Description
`-verbose`	✓	Show offending objects rather than just error counts.
`-unconstrained_endpoints`	✓	Check path endpoints for timing constraints (timing check or set_output_delay).
`-multiple_clock`	✓	Check register/latch clock pins for multiple clocks.
`-no_clock`	✓	Check register/latch clock pins for a clock.
`-no_input_delay`	✓	Check for inputs that do not have a set_input_delay command.
-no_output_delay	✓	Check for outputs that do not have a set_output_delay command.
`-loops`	✓	Check for combinational logic loops.
`-generated_clocks`	✓	Check that generated clock source pins have been defined as clocks.
`> filename`	✓	Write output to file.
`>> filename`	✓	Append output to file.

Example

To check the effectiveness of the timing constraint to fully constrain all of the design's timing endpoints, the check_setup command can be run after [run_prepare]:

The following generates standard output summarizing any checks that violate:

```
check_setup
```

The following reports only a summary of the "unconstrained endpoints":

```
check_setup -unconstrained_endpoints
```

The following will create an "check_setup.rpt" file, and will indicate all of the information available for each violations:

```
check_setup -unconstrained_endpoints -verbose > check_setup.rpt
check_setup -multiple_clock -verbose >> check_setup.rpt
check_setup -no_clock -verbose >> check_setup.rpt
check_setup -no_input_delay -verbose >> check_setup.rpt
check_setup -no_output_delay -verbose >> check_setup.rpt
check_setup -loops -verbose >> check_setup.rpt
check_setup -generated_clocks -verbose >> check_setup.rpt
```

prepare_sta

The prepare_sta command prepares the ACE Static Timing Analyzer (STA) for interactive use. This step is required before other interactive timer commands can be used. Typically, prepare_sta is only used after place and route. If the netlist, or the placement or routing, is modified, run this command again before interactive timing commands are used.

```
Command Syntax

prepare_sta (-slowc | -fastc) [-unconstrained]
```

Table 154: *Command-line Options for prepare_sta*

Argument	Optional	Description
-slowc		Use delays for the slow timing corner. This option is often used for verifying setup time requirements. (1)
-fastc		Use delays for the fast timing corner. This option is often used for verifying hold time requirements. (1)
-unconstrained	✓	Enable reporting of unconstrained paths.

Table Notes

1. Exactly one timing corner must be specified.

Example

To change the Tcl interface to be in STA interactive mode to analyze the slow timing arcs, enter the following from the ACE Tcl window, while in the ACE Tcl interface mode:

```
prepare_sta -slowc
```

When ACE is in STA "slowc" mode, to look at the fast timing arcs, enter the following:

```
reset_sta
prepare_sta -fastc
```

report_checks

The `report_checks` command reports the timing results for paths in the design.

```
Command Syntax

report_checks [-from <list>] [-to <list>] [-rise_to <list>] [-fall_to <list>] [-path_delay <min,max>] [-group_count <int>] [-endpoint_count <int>] [-through <list>] [-rise_through <list>] [-fall_through <list>] [-slack_max <float>] [-slack_min <float>] [-sort_by_slack <string>] [-path_group <list>] [-format <end,full,short,summary>] [-fields <list: input_pins,nets>] [-digits <int>] [-no_line_split]
```

Table 155: *Command Line Options for report_checks*

Argument	Optional	Description
`-from <list>`	✓	Report only paths starting at the specified objects: clocks, instances, ports, or pins.
`-to <list>`	✓	Report only paths ending at the specified objects: clocks, instances, ports, or pins.
`-rise_to <list>`	✓	Report only paths ending rising edge at the specified objects: clocks, instances, ports, or pins.
`-fall_to <list>`	✓	Report only paths ending falling edge at the specified objects: clocks, instances, ports, or pins.
`-path_delay <min,max>`	✓	The type of timing analysis. Currently only `max` (for setup analysis) and `min` (for hold time analysis) are supported. The default is `max`.
`-group_count <int>`	✓	The maximum number of paths to report, per clock group.
`-endpoint_count <int>`	✓	The number of paths to report per endpoint (default 1).
`-through <list>`	✓	Report only paths through the specified objects: instances, pins, or nets.

Argument	Optional	Description
`-rise_through <list>`	✓	Report only paths through a rising edge at the specified objects: instances, pins, or nets.
`-fall_through <list>`	✓	Report only paths through a falling edge at the specified objects: instances, pins, or nets.
`-slack_max <float>`	✓	Report only paths with slack less than this number.
`-slack_min <float>`	✓	Report only paths with slack larger than this number.
`-sort_by_slack <string>`	✓	Specifies sort order by timing "slack".
`-path_group <list>`	✓	Report only paths in these groups.
`-format <type>`	✓	Specifies which format to report for each path. [end, full, short, summary]
`-fields <list>`	✓	Report extra fields to the path report: List of input_pins|nets
`-digits <int>`	✓	Number of digits to print after the decimal point.
`-no_line_split`	✓	Do not break long lines.
`-unconstrained`	✓	Report unconstrained paths.
`> filename`	✓	Write output to file.
`>> filename`	✓	Append output to file.

report_clock_properties

The report_clock_properties command reports the clock defined for the timer in the design.

```
Command Syntax

 report_clock_properties
```

Table 156: ***Command-line Options for report_clock_properties***

Argument	Optional	Description
`> filename`	✓	Write output to file.
`>> filename`	✓	Append output to file.

Example

```
cmd> report_clock_properties

Clock  Period Waveform
----------------------------------------------------
clk[0] 2.500  0.000 1.250
clk[1] 2.500  0.000 1.250
clk[2] 2.500  0.000 1.250
clk[3] 2.500  0.000 1.250
clk[4] 2.500  0.000 1.250
clk[5] 2.500  0.000 1.250
clk[6] 2.500  0.000 1.250
clk[7] 2.500  0.000 1.250
clk[8] 2.500  0.000 1.250
clk[9] 2.500  0.000 1.250
```

reset_sta

The reset_sta command exits interactive timer mode. This command should be used before changing placement or routing, otherwise the STA might use stale data.

```
Command Syntax

reset_sta
```

Example

When in ACE interactive timing mode (slowc or fastc), to run non-timing related commands, enter the following:

```
cmd> reset_sta
```

To switch from interactive timing mode (slowc) to (fastc), enter the following:

```
reset_sta
prepare_sta -fastc
```

ACE Tcl Commands

The following commands are used only within ACE. These are not available within Synplify, etc.

add_clock_preroute

`add_clock_preroute <net_name> <track_list> [-clock_regions <list>] [-clusters <list>] [-placement_regions <list>] [-partitions <list>] [-data_region]`

This command will take a clock or reset net, and constrain it to be routed it over the clock tracks specified into the clock regions, fabric clusters, partition bounding boxes, or placement regions depending on what the user specifies.

Argument	Optional	Description
`<net_name>`		The name of the clock or reset net to be pre-routed.
`<track_list>`		The list of integer clock track numbers to pre-route this net on. Valid clock track numbers are device-specific.
`[-clock_regions <list>]`	Y	The list of clock region names to pre-route this net into. Valid clock region names are device-specific.
`[-clusters <list>]`	Y	The list of cluster names to pre-route this net into. Valid cluster names are device-specific.
`[-placement_regions <list>]`	Y	The list of placement region names to pre-route this net into. Valid placement region names are device-specific.
`[-partitions <list>]`	Y	The list of partition names to pre-route this net into. Valid partition are device-specific.
`[-data_region]`	Y	The -data_region option is only valid for non-clock nets. You can optionally use -data_region to route the net using region resources. "data_center" is applied by default if this option isn't used.

add_project_constraints

`add_project_constraints <file> [-project <string>] [-corner <string>] [-temperature <string>] [-voltage <string>]`

This command adds a link to an SDC, PDC, or TCL constraint file to a project.

Argument	Optional	Description
`<file>`		The required <file> argument is used to specify the file path to the SDC, PDC, or TCL constraint file.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the SDC constraint file to be added to.
`[-corner <string>]`	Y	The optional -corner <corner> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given process corner. Valid values are "fast" and "slow"
`[-temperature <string>]`	Y	The optional -temperature <temp> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given temperature corner. Valid values are device-specific and must match a value from the junction_temperature impl option list.
`[-voltage <string>]`	Y	The optional -voltage <v> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given core voltage corner. Valid values are device-specific and must match a value from the core_voltage impl option list.

After a project has been created, you can point to a constraint file (SDC or PDC) using the following command. In this example, there is an existing file located at `../constraints/top.sdc`:

```
add_project_constraints -project [get_active_project] "../constraints/top.sdc"
```

Also See

`create_project`

`get_active_project`

add_project_ip

`add_project_ip <list_of_files> [-project <string>]`

This command associates one or more IP settings files with a project.

Argument	Optional	Description
`<list_of_files>`		The required <list_of_files> argument is used to specify the file paths to the IP settings files. The file paths may be absolute, or may be relative to the acxprj file's directory.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be added to. The named project must already be opened in ACE.

add_project_netlist

`add_project_netlist <file> [-project <string>] [-impl <string>]`

This command adds a link to a verilog netlist file to a project.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the verilog netlist.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be added to.
[-impl <string>]	Y	The optional -impl <implName> options is used to select an alternate netlist for a different impl, i.e. due to varied synthesis tools or options.

add_region_find_insts

`add_region_find_insts <region> <find_command> [-flops_only] [-clocks_only] [-include_constants] [-batch] [-verbose]`

Add user design instances to a placement region constraint using a find command

Argument	Optional	Description
<region>		Name of the region
<find_command>		Find command used to get list of user design instances
[-flops_only]	Y	When adding instances, filter out all instances except flops
[-clocks_only]	Y	When adding instances, filter out all instances with no connected clock
[-include_constants]	Y	When adding instances, do not filter out power/ground constants
[-batch]	Y	Postpone application of this constraint until apply_placement is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.
[-verbose]	Y	Print additional command status messages.

add_region_insts

`add_region_insts <region> <insts> [-flops_only] [-clocks_only] [-include_constants] [-batch] [-verbose]`

Add user design instances to a placement region constraint

Argument	Optional	Description
<region>		Name of the region

Argument	Optional	Description
`<insts>`		List of user design instances
`[-flops_only]`	Y	When adding instances, filter out all instances except flops
`[-clocks_only]`	Y	When adding instances, filter out all instances with no connected clock
`[-include_constants]`	Y	When adding instances, do not filter out power/ground constants
`[-batch]`	Y	Postpone application of this constraint until apply_placement is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.
`[-verbose]`	Y	Print additional command status messages.

apply_highlights

`apply_highlights [-insts] [-nets] [-paths]`

This command updates the GUI with highlighting information on the present design.

Argument	Optional	Description
`[-insts]`	Y	Update highlighting of all instances in the design
`[-nets]`	Y	Update highlighting of all nets in the design
`[-paths]`	Y	Update highlighting of all paths in the design

apply_placement

`apply_placement [-batch] [-defparams] [-partition]`

Apply batched pre-placement commands

Argument	Optional	Description
`[-batch]`	Y	Specifies that placement should be applied from batched placement commands
`[-defparams]`	Y	Specifies that placement should be applied from defparams
`[-partition]`	Y	Specifies that placement should be applied on partition anchor instances before calling move_partition

check_project_status

`check_project_status`

This command checks if any project source files have changed since running the prepare flow step on the active implementation. If the source files are consistent, no message is printed. Otherwise, warnings are printed for each out of sync file.

clean_project

`clean_project [-project <string>] [-impl_names <list>]`

This command deletes output files from multiple implementations on the file system. The implementations' output directories on the file system are not deleted.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) from which the implementations' output files will be removed. If no projectName is specified, the active project will be used by default.
`[-impl_names <list>]`	Y	The optional -impl_names <list> argument is used to specify the names of the implementations to remove. If no list is specified, the output files for all implementations under the project will be deleted.

clear_arcs

`clear_arcs [-id <int>]`

This command allows you to clear a custom arc or all the arcs on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single arc to clear. If this option is not used, all arcs will be cleared.

clear_drawing

`clear_drawing`

This command clears the current custom drawing on the GUI's Floorplanner view.

clear_flow

`clear_flow`

This command clears user design DB and the completion status of all flow steps.

clear_lines

`clear_lines [-id <int>]`

This command allows you to clear a custom line or all the lines on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single line to clear. If this option is not used, all lines will be cleared.

clear_ovals

`clear_ovals [-id <int>]`

This command allows you to clear a custom oval or all the ovals on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single oval to clear. If this option is not used, all ovals will be cleared.

clear_polygons

`clear_polygons [-id <int>]`

This command allows you to clear a custom polygon or all the polygons on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single polygon to clear. If this option is not used, all polygons will be cleared.

clear_rectangles

`clear_rectangles [-id <int>]`

This command allows you to clear a custom rectangle or all the rectangles on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single rectangle to clear. If this option is not used, all rectangles will be cleared.

clear_strings

`clear_strings [-id <int>]`

This command allows you to clear a custom string or all the strings on the GUI's Floorplanner view.

Argument	Optional	Description
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for a single string to clear. If this option is not used, all strings will be cleared.

clock_info

```
clock_info [-domain <string>] [-pin <string>] [-net <string>] [-all] [-unique] [-multi]
[-freq] [-period] [-phase] [-edge_type] [-routing_props] [-core] [-driver] [-clock_net]
[-is_clock] [-info] [-group] [-equal] [-names] [-sdc]
```

Return information from the clock database. If a domain is specified, by default the name of the domain is returned. If no domain is specified, by default a list of domains is returned. Options may modify the type of value that is returned.

Argument	Optional	Description
`[-domain <string>]`	Y	specifies name of domain
`[-pin <string>]`	Y	use the domain of this pin
`[-net <string>]`	Y	use the domain of this net
`[-all]`	Y	even report uninteresting domains
`[-unique]`	Y	use a unique domain for domains with the same frequency
`[-multi]`	Y	with -net or -pin: report a list of domains instead of a single domain
`[-freq]`	Y	return the frequency (MHz); requires a domain
`[-period]`	Y	return the period (ps); requires a domain
`[-phase]`	Y	return the phase; requires a domain
`[-edge_type]`	Y	1 for pos-edge, -1 for neg-edge, 0 for combinational; requires a domain
`[-routing_props]`	Y	list of strings denoting the routing properties (if set); requires a domain
`[-core]`	Y	1 if used in the core, otherwise 0; or list of domains used in core
`[-driver]`	Y	name of the driving pin or port; requires a domain
`[-clock_net]`	Y	name of the clock net; requires a domain
`[-is_clock]`	Y	true if net is a clock net; requires a pin or net
`[-info]`	Y	list as for 'array set'; requires a domain
`[-group]`	Y	list of related domains, or list of groups
`[-equal]`	Y	list of domains with same frequency; requires a domain
`[-names]`	Y	list of all names for the domain; requires a domain
`[-sdc]`	Y	return list of sdc commands

clock_relation

```
clock_relation <domain1> <domain2> [-default] [-group] [-sdc]
```

Return relation between clocks. For related clocks the return is a list with 5 values: the word "related" followed by T1 T2 e1 e2. T is the abstract period: T1/T2 = period1/period2. e is an abstract offset (in the same units as T). By default the numbers are as small as possible, but with -group all related clocks use the same units.

Argument	Optional	Description
`<domain1>`		first domain
`<domain2>`		second domain
`[-default]`	Y	apply current default_relation rule
`[-group]`	Y	values are in group units
`[-sdc]`	Y	return list of sdc commands

create_boundary_pins

```
create_boundary_pins <name> <boundary_pin_names> [-clock] [-data] [-purpose <string>]
```

This command instantiates IPIN/OPINs at the Core/IO Ring boundary

Argument	Optional	Description
`<name>`		A reference to a net in the design where the boundary pins should be inserted (<p: toplevel_portname> | <t:user_pin_name> | <n:net_name>)
`<boundary_pin_names>`		A list of one or more instance names for the boundary pins to be inserted on the given net
`[-clock]`	Y	Create a clock pin even if the specified net is a data net
`[-data]`	Y	Create a data pin even if the specified net is a clock net
`[-purpose <string>]`	Y	Set the PURPOSE property on the created boundary pin instances. Legal values are: 'USER' (the default), 'JTAG', 'CFG'.

create_equivalent_regions

```
create_equivalent_regions <source>
```

Create non-overlapping placement regions which have the same tiles in the same order as the provided <source>

Argument	Optional	Description
`<source>`		Name of the source. May be a region or a partition name.

create_flow_step

```
create_flow_step <id> <label> [-command <string>] [-parent_id <string>] [-required] [-
skip_for_eval_mode] [-offset <int>] [-description <string>]
```

This command creates a flow step definition, which is basically a wrapper around an existing command or script that manages flow status and dependencies.

Argument	Optional	Description
`<id>`		The required <id> string argument specifies the identifier of the flow step to create. The <id> argument must be unique among all flow step ids in ACE.
`<label>`		The required <label> argument specifies the label string to display in the GUI for this flow step. The label should be as short as possible.
`[-command <string>]`	Y	The optional -command <command> option specifies the TCL command to run when this flow step is invoked.
`[-parent_id <string>]`	Y	The optional -parent_id <parentId> option specifies the flow step id of an existing flow step (which does not have a command of its own) that this new flow step will be grouped under in the flow hierarchy.
`[-required]`	Y	The optional -required option specifies whether or not this flow step is required for further processing of the flow. If this option is not used, the user may optionally enable or disable this flow step for use in running the flow with run_flow.
`[-skip_for_eval_mode]`	Y	The optional -skip_for_eval_mode option specifies whether or not this flow step will be skipped when flow_mode is set to evaluation.
`[-offset <int>]`	Y	The optional -offset <offset> option specifies the position (as a positive integer) under the parent flow step (or top level) at which this flow step should be inserted. Without this option, the flow step will be appended to the end of the flow steps under the parent flow step (or top level).
`[-description <string>]`	Y	The optional -description <description> option specifies the description text to display in the GUI for this flow step.

create_impl

```
create_impl <implName> [-project <string>] [-copy] [-not_active]
```

This command creates a new implementation in a project. This command causes the new implementation to become the active implementation.

Argument	Optional	Description
`<implName>`		The required <implName> argument is used to specify the name for the new implementation.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the implementation to be added to.
`[-copy]`	Y	The optional -copy option is used to copy the implementation options of the active implementation into the newly created implementation.
`[-not_active]`	Y	If this option is set, the new project impl will not be activated and the active impl in the current ACE session will not be changed.

create_path

```
create_path <pins> [-id <string>] [-rgb <list>]
```

This command creates a user-defined pin path that may be used for selection or highlighting.

Argument	Optional	Description
`<pins>`		The required <pins> list argument specifies the ordered list of instance pins.
`[-id <string>]`	Y	The optional -id <id> option specifies the string id to use for this path. If an id is not specified, a unique id will be automatically generated.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option is used to specify the RGB (Red-Green-Blue) color value to use for highlighting the specified objects as a 3 element list of integers {red green blue}. If the -rgb option is not used, then the objects in the list will be un-highlighted.

create_project

```
create_project <projectFile> [-impl <string>] [-not_active]
```

This command creates a new project in ACE.

Argument	Optional	Description
`<projectFile>`		The required <projectFile> argument is used to specify the project file location for the new project. The file name is used as the project's name in ACE.
`[-impl <string>]`	Y	The optional -impl <implName> option is used to specify the name of the initial implementation for this new project.
`[-not_active]`	Y	If this option is set, the new project impl will not be activated and the active impl in the current ACE session will not be changed.

create_region

```
create_region <name> <bounds> [-find_insts <string>] [-insts <list>] [-
snap_to_clock_regions] [-snap_to_fabric_clusters] [-snap <string>] [-soft] [-type
<string>] [-flops_only] [-clocks_only] [-include_routing] [-include_constants] [-
pr_zone] [-batch] [-verbose]
```

This command creates a placement region in the Core with the given name and bounding box of tiles. Instances may be added to this placement region to create a region constraint for the placer.

Argument	Optional	Description
`<name>`		Name of the region
`<bounds>`		List of bounding box coordinates {x1 y1 x2 y2}. x1 and y1 are the upper left corner of the box. x2 and y2 are the lower right corner of the box.

Argument	Optional	Description
`[-find_insts <string>]`	Y	Pass in a find command string to create the list of user design instances to constrain into this region's bounding box for the placer.
`[-insts <list>]`	Y	List of user design instances to constrain into this region's bounding box for the placer.
`[-snap_to_clock_regions]`	Y	Snap the bounding box to clock region boundaries (deprecated, use '-snap clock_regions')
`[-snap_to_fabric_clusters]`	Y	Snap the bounding box to fabric cluster boundaries (deprecated, use '-snap fabric_clusters')
`[-snap <string>]`	Y	How to snap the region bounding box coordinates. Legal values are: 'none', 'tiles' (the default), 'fabric_clusters', 'clock_regions'.
`[-soft]`	Y	Create a 'soft' placement region, which attempts to pull instance placement to its center, but allows instance placement to overflow the bounds of the region (deprecated, use '-type soft')
`[-type <string>]`	Y	What type of placement region this is. Legal values are: 'inclusive' (the default), 'keepout', 'soft'. Instances added to an 'inclusive' region (and attached routing wires when '-include_routing' is set) will be placed within the region bounding box. An 'inclusive' region permits instances to be placed inside the region even if they do not belong to the region. A 'keepout' region prevents any instances (and routing wires when '-include_routing is set) from being placed inside the region. No instances may be added to a 'keepout' region. Instances added to an 'soft' region will be pulled toward the region's center during placement, but instances are permitted to overflow the bounds of the 'soft' region.
`[-flops_only]`	Y	When adding instances, filter out all instances except flops
`[-clocks_only]`	Y	When adding instances, filter out all instances that do not have a connected clock pin
`[-include_routing]`	Y	Constrain routing wires, as well as instances, to stay within the region boundary box
`[-include_constants]`	Y	When adding instances, do not filter out power/ground constants
`[-pr_zone]`	Y	This will indicate that the placement region is intended to be used for partial reconfiguration.
`[-batch]`	Y	Postpone application of this constraint until apply_placement is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.

Argument	Optional	Description
[-verbose]	Y	Print additional command status messages.

deselect

`deselect [-objects <list>]`

This command removes objects from the current list of selected objects.

Argument	Optional	Description
[-objects <list>]	Y	The optional -objects <objects> option is used to specify a list of objects to remove from the current selection. Objects must be prepended with object type prefixes (see "find" command). Objects in the <objects> list that are not in the current selection are silently ignored. Without this option, the deselect command will remove all objects from the current selection.

disable_flow_step

`disable_flow_step <id>`

This command disables an existing optional flow step from being run during a "run" command.

Argument	Optional	Description
<id>		The required <id> argument specifies the id of the flow step to disable.

disable_project_constraints

`disable_project_constraints [-project <string>] [-impl <string>] <file>`

This command disables project constraints files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to disable constraints for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to disable constraints for.
<file>		The project constraints file to disable for a project implementation.

display_file

`display_file <file> [-line_number <int>]`

This command automatically opens a file in the GUI. This command has no effect in batch mode.

Argument	Optional	Description
`<file>`		The required <file> argument specifies the path to the file to automatically open in the GUI (when in -gui mode).
`[-line_number <int>]`	Y	The optional -line_number option allows you to open a text file to a particular line.

display_netlist

`display_netlist <object>`

This command attempts to open the gate level netlist file in the GUI for the given user design instance or net. This command has no effect when ACE is running in batch mode.

Argument	Optional	Description
`<object>`		The required <object> argument specifies the instance (i:) or net (n:) name for which the netlist file will be opened in the GUI.

display_properties

`display_properties <object> [-print]`

This command displays detailed properties of the specified object in the GUI, and optionally prints the details to the console.

Argument	Optional	Description
`<object>`		The required <object> argument specifies the object to get properties for.
`[-print]`	Y	The -print option will print all the object property details to the TCL console in addition to sending the data to the GUI.

draw_arc

`draw_arc <x> <y> <width> <height> <startAngle> <arcAngle> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]`

This command allows you to draw a custom arc on the GUI's Floorplanner view.

Argument	Optional	Description
`<x>`		The required <x> argument specifies the upper-left x coordinate for the arc.
`<y>`		The required <y> argument specifies the upper-left y coordinate for the arc.
`<width>`		The required <width> argument specifies the width of the arc.
`<height>`		The required <height> argument specifies the height of the arc.

Argument	Optional	Description
`<startAngle>`		The required <startAngle> argument specifies the starting angle of the arc.
`<arcAngle>`		The required <arcAngle> argument specifies the angle of the arc.
`[-layer <int>]`	Y	The optional -layer <layer> option specifies the drawing layer for the arc. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for the arc. If this option is not used, a unique id will be automatically generated and returned by the command.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option specifies the rgb color value for the arc as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
`[-batch]`	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
`[-thickness <int>]`	Y	The optional -thickness <pixels> option specifies the arc thickness in pixels. If this option is not used, a thickness of 1 will be used.
`[-fill]`	Y	The optional -fill option specifies whether the arc should be filled with color or not. If this option is not used, the arc will be hollow.

draw_line

```
draw_line <x1> <y1> <x2> <y2> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-
thickness <int>]
```

This command allows you to draw a custom line on the GUI's Floorplanner view.

Argument	Optional	Description
`<x1>`		The required <x1> argument specifies the first x coordinate for the line.
`<y1>`		The required <y1> argument specifies the first y coordinate for the line.
`<x2>`		The required <x2> argument specifies the second x coordinate for the line.
`<y2>`		The required <y2> argument specifies the second y coordinate for the line.
`[-layer <int>]`	Y	The optional -layer <layer> option specifies the drawing layer for the line. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for the line. If this option is not used, a unique id will be automatically generated and returned by the command.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option specifies the rgb color value for the line as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.

Argument	Optional	Description
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the line thickness in pixels. If this option is not used, a thickness of 1 will be used.

draw_oval

`draw_oval <x> <y> <width> <height> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]`

This command allows you to draw a custom oval on the GUI's Floorplanner view.

Argument	Optional	Description
<x>		The required <x> argument specifies the upper-left x coordinate for the oval.
<y>		The required <y> argument specifies the upper-left y coordinate for the oval.
<width>		The required <width> argument specifies the width of the oval.
<height>		The required <height> argument specifies the height of the oval.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the oval. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the oval. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the oval as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the oval thickness in pixels. If this option is not used, a thickness of 1 will be used.
[-fill]	Y	The optional -fill option specifies whether the oval should be filled with color or not. If this option is not used, the oval will be hollow.

draw_polygon

`draw_polygon <points> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]`

This command allows you to draw a custom polygon on the GUI's Floorplanner view.

Argument	Optional	Description
`<points>`		The required <points> argument specifies the list of x-y coordinates for polygon, starting with the x coordinate and alternating. For example: {1 1 2 2 3 5 1 6}.
`[-layer <int>]`	Y	The optional -layer <layer> option specifies the drawing layer for the arc. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for the arc. If this option is not used, a unique id will be automatically generated and returned by the command.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option specifies the rgb color value for the polygon as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
`[-batch]`	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
`[-thickness <int>]`	Y	The optional -thickness <pixels> option specifies the arc thickness in pixels. If this option is not used, a thickness of 1 will be used.
`[-fill]`	Y	The optional -fill option specifies whether the arc should be filled with color or not. If this option is not used, the arc will be hollow.

draw_rectangle

```
draw_rectangle <x> <y> <width> <height> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]
```

This command allows you to draw a custom rectangle on the GUI's Floorplanner view.

Argument	Optional	Description
`<x>`		The required <x> argument specifies the upper-left x coordinate for the rectangle.
`<y>`		The required <y> argument specifies the upper-left y coordinate for the rectangle.
`<width>`		The required <width> argument specifies the width of the rectangle.
`<height>`		The required <height> argument specifies the height of the rectangle.
`[-layer <int>]`	Y	The optional -layer <layer> option specifies the drawing layer for the rectangle. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for the rectangle. If this option is not used, a unique id will be automatically generated and returned by the command.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option specifies the rgb color value for the rectangle as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.

Argument	Optional	Description
`[-batch]`	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
`[-thickness <int>]`	Y	The optional -thickness <pixels> option specifies the rectangle thickness in pixels. If this option is not used, a thickness of 1 will be used.
`[-fill]`	Y	The optional -fill option specifies whether the rectangle should be filled with color or not. If this option is not used, the rectangle will be hollow.

draw_string

`draw_string <x> <y> <string> [-layer <int>] [-id <int>] [-rgb <list>] [-batch]`

This command allows you to draw a custom string on the GUI's Floorplanner view.

Argument	Optional	Description
`<x>`		The required <x> argument specifies the x coordinate for the string.
`<y>`		The required <y> argument specifies the y coordinate for the string.
`<string>`		The required <string> argument specifies the string text.
`[-layer <int>]`	Y	The optional -layer <layer> option specifies the drawing layer for the string. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
`[-id <int>]`	Y	The optional -id <id> option specifies a unique id for the string. If this option is not used, a unique id will be automatically generated and returned by the command.
`[-rgb <list>]`	Y	The optional -rgb <rgb> option specifies the rgb color value for the string as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
`[-batch]`	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.

enable_flow_step

`enable_flow_step <id>`

This command enables an existing optional flow step to be run during a "run" command.

Argument	Optional	Description
`<id>`		The required <id> argument specifies the id of the flow step to enable.

enable_project_constraints

`enable_project_constraints [-project <string>] [-impl <string>] <file>`

This command enables project constraints files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to enable constraints for.
`[-impl <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to enable constraints for.
`<file>`		The project constraints file to enable for a project implementation.

export_all_partitions

```
export_all_partitions [-info_list]
```

Command to export the place-and-route database and blackbox Verilog model for all leaf-level partitions in the design

Argument	Optional	Description
`[-info_list]`	Y	Return a Tcl list containing {<partition> <view> <epdb filename> <blackbox filename>} for each partition

export_partition

```
export_partition <partition> [-dboutputfile <string>] [-bboutputfile <string>] [-
info_list]
```

Command to export the place-and-route database and blackbox Verilog model for a partition

Argument	Optional	Description
`<partition>`		Export the place-and-route database and blackbox Verilog model for the specified partition
`[-dboutputfile <string>]`	Y	Specifies the output file name for the partition database (default is <active_impl_dir> /output/partitions/<cellname>_<partition>.epdb)
`[-bboutputfile <string>]`	Y	Specifies the output file name for the partition blackbox Verilog model (default is <active_impl_dir>/output/blackboxes/<cellname>_bb.v)
`[-info_list]`	Y	Return a Tcl list containing {<partition> <view> <epdb filename> <blackbox filename>} for each partition

filter

```
filter <objects> [-patterns <list>] [-insts] [-nets] [-ports] [-pins] [-paths] [-sites]
[-filter <string>] [-no_prefix]
```

This command takes a TCL list of DB objects and returns a filtered TCL list of objects that match the filter options passed in. Each object name in the returned list is prepended with an object type indicator (unless -no_prefix is used). Object types prefixes are: p: = port (top level user design), t: = pin, i: = instance, n: = net. Find results may contain a mixture of object types. The -insts, -nets, -ports, and -pins object type options may be used to filter the results to just those object types. Specifying no object type options will result in a search of all object types.

Argument	Optional	Description
`<objects>`		The required <objects> argument specifies a list of object names to filter.
`[-patterns <list>]`	Y	The optional -patterns argument specifies a list of pattern strings to match object names against. Each pattern string in the list may use '*' and '?' wildcard characters for matching.
`[-insts]`	Y	The optional -insts object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -insts option is not used, then the results will not contain any instance objects.
`[-nets]`	Y	The optional -nets object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -nets option is not used, then the results will not contain any net objects.
`[-ports]`	Y	The optional -ports object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -ports option is not used, then the results will not contain any top level user design port objects.
`[-pins]`	Y	The optional -pins object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -pins option is not used, then the results will not contain any pin objects.
`[-paths]`	Y	The optional -paths object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -paths option is not used, then the results will not contain any path objects.
`[-sites]`	Y	The optional -sites object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -sites option is not used, then the results will not contain any site objects.
`[-filter <string>]`	Y	The optional -filters option may be used to specify a boolean expression of object properties to filter the results with. Each property filter in the expression must follow the filter syntax of @<propertyname><operator><value>. Multiple property filters may be used in the expression by using boolean operators. For example: "find * -filter {@type=DFF || @type=LUT4}". The supported filter property names are currently: @async_reset, @attribute, @clock, @clock_as_data, @clock_domain, @clock_region, @data_as_clock, @direction, @driver_type, @driving_net, @driving_pin, @enable, @fanout, @fixed_placement, @partition, @placed, @power, @power_rank, @region, @reset, @sink_type, or @type. The supported filter operators are currently: >, <, !, and =. The supported boolean operators (when using multiple filters) are currently: &&, ||, and ==.

Argument	Optional	Description
`[-no_prefix]`	Y	The optional -no_prefix option is used to remove the object type prefix from the names returned in the results.

See also: Object Type Prefixes (see page 305), Search Filter Builder Dialog (see page 174), Filter Properties (see page 244), `find`, Search View. (see page 132)

find

```
find <patterns> [-insts] [-nets] [-ports] [-pins] [-paths] [-sites] [-filter <string>] [-sort <string>] [-sort_order <string>] [-no_prefix] [-no_refresh] [-handle] [-warning] [-error]
```

This command returns a TCL list of object names that match any of the pattern strings passed in. Each object name in the returned list is prepended with an object type indicator (unless -no_prefix is used). Object types prefixes are: p: = port (top level user design), t: = pin, i: = instance, n: = net. Find results may contain a mixture of object types. The -insts, -nets, -ports, and -pins object type options may be used to filter the results to just those object types. Specifying no object type options will result in a search of all object types.

Argument	Optional	Description
`<patterns>`		The required <patterns> argument specifies a list of pattern strings to match object names against. The pattern for matching with specific pins must have separator '/' in the form of <instance pattern>/<pin pattern>. Each pattern string in the list may use '*' and '?' wildcard characters for matching.
`[-insts]`	Y	The optional -insts object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -insts option is not used, then the results will not contain any instance objects.
`[-nets]`	Y	The optional -nets object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -nets option is not used, then the results will not contain any net objects.
`[-ports]`	Y	The optional -ports object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -ports option is not used, then the results will not contain any top level user design port objects.
`[-pins]`	Y	The optional -pins object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -pins option is not used, then the results will not contain any pin objects.
`[-paths]`	Y	The optional -paths object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -paths option is not used, then the results will not contain any path objects.

Argument	Optional	Description
[-sites]	Y	The optional -sites object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -sites option is not used, then the results will not contain any site objects.
[-filter <string>]	Y	The optional -filters option may be used to specify a boolean expression of object properties to filter the results with. Each property filter in the expression must follow the filter syntax of @<propertyname><operator><value>. Multiple property filters may be used in the expression by using boolean operators. For example: "find * -filter {@type=DFF || @type=LUT4}". The supported filter property names are currently: @async_reset, @attribute, @clock, @clock_as_data, @clock_domain, @clock_region, @data_as_clock, @direction, @driver_type, @driving_net, @driving_pin, @enable, @fanout, @fixed_placement, @partition, @placed, @power, @power_rank, @region, @reset, @sink_type, or @type. The supported filter operators are currently: >, <, !, and =. The supported boolean operators (when using multiple filters) are currently: &&, ||, and ==.
[-sort <string>]	Y	The -sort option allows the user to specify the type of sort performed on the find results list. The default is "dictionary". Other options are "ascii" or "none"
[-sort_order <string>]	Y	The -sort_order option allows the user to specify the direction of sort performed on the find results list. You may specify either "increasing" or "decreasing". The default is "increasing".
[-no_prefix]	Y	The optional -no_prefix option is used to remove the object type prefix from the names returned in the results
[-no_refresh]	Y	The optional -no_refresh option is used to prevent sending an update to the GUI Search View to optimize speed
[-handle]	Y	The optional -handle option is used to return the reserve string "@@FindResults" instead of the TCL list of object names. This handle can then be used in the highlight command to speed up processing by avoiding extra name parsing
[-warning]	Y	Print warning message if the find command does not find any objects.
[-error]	Y	Print message and error out if the find command does not find any objects.

The ACE GUI provides a graphical interface for the `find` command through the Search View (see page 132). See also: Object Type Prefixes (see page 305), Search Filter Builder Dialog (see page 174), Filter Properties (see page 244), `filter`, `select`, Selection View (see page 136), `trace_connections`. (see page 607)

generate_ioring_design_files

```
generate_ioring_design_files <outputDir> [-add_to_project]
```

This command generates the all IO Ring design files for the active ACE project, using all ACXIP files that have been added to the active project.

Argument	Optional	Description
`<outputDir>`		The required <outputDir> argument allows the user to specify the directory path to output the IO Ring design files into.
`[-add_to_project]`	Y	Using the -add_to_project option automatically adds the required generated IO Ring design files to your ACE project. This includes utilization XML, SDC constraints, PDC constraints, and IO Ring bitstream files.

generate_ip_design_files

`generate_ip_design_files <acxipFile>`

This command generates the enabled design files for a given IP configuration (.acxip file).

Argument	Optional	Description
`<acxipFile>`		The required <acxipFile> argument specifies the IP configuration (.acxip file) to generate design files for.

generate_route_delay_table

`generate_route_delay_table [-outputfile <string>]`

This command extracts route delay numbers on nets for estimating the cell-cell route delays vs fanout.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation .debug directory and is named <design_name>_route_delay.log.

get_ace_cputime

`get_ace_cputime`

This command returns the cumulative cpu time of this ACE process

get_ace_current_memory_usage

`get_ace_current_memory_usage`

This command returns the current memory usage (in kB) of this ACE process

get_ace_ext_dir

`get_ace_ext_dir`

This command returns the path to the ACE Extensions directory if one has been enabled.

get_ace_ext_lib

`get_ace_ext_lib <partName>`

This command returns the blackbox library path in the ACE Extensions directory for the specified partname if one has been enabled.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the target device to find the blackbox library file for. The part name specified must exist among the valid part names in the ACE installation.

get_ace_peak_memory_usage

`get_ace_peak_memory_usage`

This command returns the maximum memory usage (in kB) of this ACE process during the current session

get_ace_version

`get_ace_version [-buildid] [-builddate] [-full]`

This command returns the version of ACE

Argument	Optional	Description
`[-buildid]`	Y	The optional -buildid option will return the buildid.
`[-builddate]`	Y	The optional -builddate option will say when ace was built.
`[-full]`	Y	The optional -full option will return the full ACE version and build designation.

get_active_impl

`get_active_impl [-quiet]`

This command returns the name of the active implementation in the current ACE session.

Argument	Optional	Description
`[-quiet]`	Y	do not print a message if there is no active project

Example

To automatically set the value of the `set_impl_option -impl` option, after the `create_project -impl` command has been run, which defines the name of the active impl, the following command can be used:

```
set_impl_option -project [get_active_project] -impl [get_active_impl]  "partname"  "AC16tSC01HI01C"
```

Also See

`create_project`

`get_active_project`

`set_impl_option`

get_active_project

`get_active_project [-quiet] [-path]`

This command returns the name of the active project (which contains the active implementation) in the current ACE session.

Argument	Optional	Description
`[-quiet]`	Y	do not print a message if there is no active project
`[-path]`	Y	Return the file path to the active project's acxprj project file, instead of the project name

get_best_multiprocess_impl

`get_best_multiprocess_impl`

This command finds the best impl from the MultiProcess Summary Report in the active project.

get_clock_region_bounds

`get_clock_region_bounds <region>`

Returns the bounding box for a clock region

Argument	Optional	Description
`<region>`		Name of the region

get_clock_regions

`get_clock_regions`

Returns the list of clock region names for the device

get_clock_type

`get_clock_type <clock>`

Get properties of a clock. For a non-driving (target) clock pin, this is a combination of local properties and properties of the clock domain.

Argument	Optional	Description
`<clock>`		net or pin ('inst/pin')

get_compatible_ordering_codes

`get_compatible_ordering_codes`

This command returns a list of compatible ordering codes for the active project based on it's device, package, and speed grade.

get_compatible_placements

`get_compatible_placements <source> [-anchor <string>] [-outputfile <string>]`

Get a list of compatible placements for the given <source> partition

Argument	Optional	Description
`<source>`		Name of the source partition
`[-anchor <string>]`	Y	Name of the anchor instance. An anchor instance will be chosen automatically if not given.
`[-outputfile <string>]`	Y	Name of optional output file. If given, the compatible placements will be written out as a series of set_placement commands.

get_current_design

`get_current_design [-quiet]`

This command returns the name of the top module in the current design. This command returns an error if no current design is loaded.

Argument	Optional	Description
`[-quiet]`	Y	do not print a message if there is no active project

get_current_partname

`get_current_partname [-quiet]`

This command returns the name of the currently loaded device.

Argument	Optional	Description
`[-quiet]`	Y	do not warn if there is no current part

get_efd_file_path

`get_efd_file_path <partName>`

This command returns the path to the efd file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the efd file for. The part name specified must exist among the valid part names in the ACE installation.

get_enabled_constraints

`get_enabled_constraints [-project <string>] [-impl <string>]`

This command returns a list of all the enabled constraint files for an implementation.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.
`[-impl <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.

get_fabricdb_path

`get_fabricdb_path <partName>`

This command returns the path to the fabric db file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the fabric db for. The part name specified must exist among the valid part names in the ACE installation.

get_file_line

`get_file_line <object>`

This command returns the file path and line offset into to the source netlist for the given user design instance or net.

Argument	Optional	Description
`<object>`		The required <object> argument specifies the instance (i:) or net (n:) name for which the file line will be retrieved.

get_flow_steps

`get_flow_steps`

This command returns a list of all the currently defined flow step id strings.

get_impl_names

`get_impl_names [-project <string>]`

This command returns a list of all the implementation names for an existing project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to get the implementation names from.

get_impl_option

`get_impl_option <option_name> [-project <string>] [-impl <string>] [-syn]`

This command returns the current value of a project implementation option. Only one option value may be retrieved at a time.

Argument	Optional	Description
<option_name>		The name of the impl option to retrieve a value for. To see a list of valid impl options, use the report_impl_options TCL command.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-syn]	Y	option is synthesis option

get_impl_option_is_supported

```
get_impl_option_is_supported <option_name> [-project <string>] [-impl <string>]
```

Returns '1' if the impl option is supported on the current device, otherwise returns '0'

Argument	Optional	Description
<option_name>		The name of the impl option to retrieve a value for. To see a list of valid impl options, use the report_impl_options TCL command.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.

get_inst_partition

```
get_inst_partition <instance>
```

Returns the Partition associated with a user design instance

Argument	Optional	Description
<instance>		Name of the instance

get_inst_region

```
get_inst_region <instance>
```

Returns the region associated with a user design instance

Argument	Optional	Description
<instance>		Name of the instance

get_installation_directory

```
get_installation_directory
```

This command returns the path to the root of the ACE installation.

get_location

```
get_location <object>
```

This command allows you to get the location of an object (i:instance, s:site, or t:pin) on the GUI's Floorplanner view.

Argument	Optional	Description
`<object>`		The required <object> argument specifies the object to get coordinates for. The correct object type prefix is required.

get_part_names

```
get_part_names
```

This command returns the list of valid part names in the installed library.

get_partition_changed

```
get_partition_changed <name>
```

Get the changed flag of a partition with the given name

Argument	Optional	Description
`<name>`		Name of the Partition

get_partition_force_changed

```
get_partition_force_changed <name>
```

Get the force changed flag of a partition with the given name

Argument	Optional	Description
`<name>`		Name of the Partition

get_partition_info

```
get_partition_info <name> [-timestamp] [-comment] [-view] [-type] [-is_import] [-
import_from] [-changed] [-id] [-disabled] [-parent] [-is_top] [-is_parent]
```

Get info of a partition with the given name

Argument	Optional	Description
`<name>`		Name of the Partition
`[-timestamp]`	Y	get timestamp

Argument	Optional	Description
[-comment]	Y	get comment
[-view]	Y	get view name
[-type]	Y	get partition type
[-is_import]	Y	was the partition imported (0 or 1)?
[-import_from]	Y	file the partition was imported from
[-changed]	Y	has the partition timestamp changed
[-id]	Y	unique partition id number
[-disabled]	Y	is the partition disabled (0 or 1)?
[-parent]	Y	name of the partition's parent partition
[-is_top]	Y	is this the top partition (0 or 1)?
[-is_parent]	Y	is this partition a parent of another (0 or 1)?

get_partition_insts

`get_partition_insts <name>`

Returns the list of user design instances in a partition

Argument	Optional	Description
<name>		Name of the Partition

get_partition_names

`get_partition_names`

Returns the list of user design partition names

get_partition_timestamp

`get_partition_timestamp <name>`

Get the timestamp of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition

get_partition_type

`get_partition_type <name>`

Get the type of a partition with the given name

Argument	Optional	Description
`<name>`		Name of the Partition

get_path_property

`get_path_property <id> [-pins] [-insts] [-nets] [-frequency] [-type] [-rgb] [-text] [-slack]`

This command returns path properties.

Argument	Optional	Description
`<id>`		The required <id> argument specifies the id of the path to get properties for.
`[-pins]`	Y	The optional -pins option returns the list of pin names that make up this path.
`[-insts]`	Y	The optional -insts option returns the list of instance names on this path.
`[-nets]`	Y	The optional -nets option returns the list of net names on this path.
`[-frequency]`	Y	The optional -frequency option returns the frequency of this path in MHz. If no frequency is defined, -1 is returned.
`[-type]`	Y	The optional -type option returns the type of path: Setup Check Met, Setup Check Violated, Hold Check Met, Hold Check Violated, Hardware Limit, or User-Defined.
`[-rgb]`	Y	The optional -rgb option returns the integer rgb highlight color value for the path. A value of -1 means it is not highlighted.
`[-text]`	Y	The optional -text option returns the details text for this path.
`[-slack]`	Y	The optional -slack option returns the slack for this path.

get_placement

`get_placement <objName>`

This command returns the site of the specified placed instance

Argument	Optional	Description
`<objName>`		The required <objName> argument is used to specify the instance or port to get placement for.

get_pod_names

`get_pod_names [-all] [-usb] [-ethernet] [-list <list>]`

Returns a list of names of available Bitporter pods.

Argument	Optional	Description
`[-all]`	Y	(default behavior) returns USB and detected Ethernet pods.
`[-usb]`	Y	(optional) returns only the USB pods.
`[-ethernet]`	Y	(optional) returns only the detected Ethernet pods.
`[-list <list>]`	Y	(optional) specifies a list of podnames whose availability should be checked.

get_project_constraint_files

`get_project_constraint_files [-project <string>]`

This command returns a list of all the constraint file paths for a project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to get the constraint file paths from.

get_project_directory

`get_project_directory [-project <string>]`

This command returns the path to a project file's parent directory

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to get the directory path from.

get_project_ip_files

`get_project_ip_files [-project <string>]`

This command returns a list of all the IP settings file paths for a project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to get the IP settings file paths from.

get_project_names

`get_project_names`

This command returns a list of all the project names loaded in the current ACE session.

get_project_netlist_files

`get_project_netlist_files [-project <string>]`

This command returns a list of all the netlist file paths for a project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to get the netlist file paths from.

get_properties

`get_properties <object>`

This command returns the list of option-value pairs for the specified object.

Argument	Optional	Description
`<object>`		The required <object> argument specifies the object to get properties for.

get_property

`get_property <object> <propName> [-object_type <string>]`

This command returns the specified property value for the specified object.

Argument	Optional	Description
`<object>`		The required <object> argument specifies which object will be queried.
`<propName>`		The required <propName> argument specifies the name of the property whose value will be retrieved.
`[-object_type <string>]`	Y	type of object: cell | pin | net | port | clock

get_pvt_corners

`get_pvt_corners [<partName>]`

This command returns a list of the valid PVT corners for the specified part

Argument	Optional	Description
`[<partName>]`	Y	Name of part

get_region_bounds

`get_region_bounds <region>`

Returns the bounding box for a placement region constraint

Argument	Optional	Description
`<region>`		Name of the region

get_region_insts

`get_region_insts <region>`

Returns the list of user design instances in a placement region constraint

Argument	Optional	Description
`<region>`		Name of the region

get_regions

`get_regions [-verbose]`

Returns the list of placement region constraint names

Argument	Optional	Description
`[-verbose]`	Y	print region information with each region

get_report_sweep_temperature_corners

`get_report_sweep_temperature_corners [-quiet]`

This command returns a list of the valid junction temperatures for the target device at the given speed grade and core voltage level

Argument	Optional	Description
`[-quiet]`	Y	do not print a message if there is no active project

get_selection

`get_selection [-insts] [-nets] [-ports] [-pins] [-paths] [-sites] [-handle]`

This command returns the current list of selected objects.

Argument	Optional	Description
`[-insts]`	Y	The optional -insts object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -insts option is not used, then the results will not contain any instance objects.
`[-nets]`	Y	The optional -nets object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -nets option is not used, then the results will not contain any net objects.

Argument	Optional	Description
[-ports]	Y	The optional -ports object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -ports option is not used, then the results will not contain any top level user design port objects.
[-pins]	Y	The optional -pins object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -pins option is not used, then the results will not contain any pin objects.
[-paths]	Y	The optional -paths object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -paths option is not used, then the results will not contain any path objects.
[-sites]	Y	The optional -sites object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -sites option is not used, then the results will not contain any site objects.
[-handle]	Y	The optional -handle option is used to return the reserve string "@@Selection" instead of the TCL list of object names. This handle can be used in the highlight command to speed up processing by avoiding extra name parsing

get_stapl_actions

`get_stapl_actions <staplfile>`

Returns a list of Actions found in the specified STAPL file, along with the Procedures making up each Action. The listed Procedures within each Action will indicate whether they are Required, Recommended (run by default, but may be disabled by the user), or Optional (not run by default, but may be enabled by the user).

Argument	Optional	Description
<staplfile>		The STAPL file whose Actions should be returned.

get_synprj_from_project

`get_synprj_from_project`

This command returns the name the of the synthesis prj-file associated with the ace project, if available.

get_techlib_name

`get_techlib_name <partName>`

This command returns the name the of black box verilog library for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlib_path

`get_techlib_path <partName>`

This command returns the path to the black box verilog library file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibdb_path

`get_techlibdb_path <partName>`

This command returns the path to the techlib db file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the techlib db for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibt_name

`get_techlibt_name <partName>`

This command returns the name of the transmuted black box verilog library for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibt_path

`get_techlibt_path <partName>`

This command returns the path to the transmuted black box verilog library file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibx_name

`get_techlibx_name <partName>`

This command returns the name the of the expanded black box verilog library for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibx_path

```
get_techlibx_path <partName>
```

This command returns the path to the expanded black box verilog library file for the given part.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

has_ace_ext_lib

```
has_ace_ext_lib <partName>
```

This command returns a 1 if the blackbox library path is configured in the ACE Extensions directory for the specified partname.

Argument	Optional	Description
`<partName>`		The required <partName> argument is used to specify the name of the target device to find the blackbox library file for. The part name specified must exist among the valid part names in the ACE installation.

has_partitions

```
has_partitions
```

Check if partitions have been defined on the design via the *.prt file

highlight

```
highlight <objects> [-rgb <list>] [-batch] [-clear]
```

This command is used to highlight or un-highlight a list of objects in the GUI's physical view.

Argument	Optional	Description
`<objects>`		The required <objects> argument is used to specify a list of objects which will have their highlight color set. Highlight of instance, net, and path object types is currently supported. All other object types passed in will be silently ignored. Objects must be prepended with object type prefixes (see "find" command).
`[-rgb <list>]`	Y	The optional -rgb <rgb> option is used to specify the RGB (Red-Green-Blue) color value to use for highlighting the specified objects as a 3 element list of 8-bit (0-255) integers {red green blue}. If the -rgb option is not used, then the objects in the list will be un-highlighted.
`[-batch]`	Y	The optional -batch option is used to suppress the refresh of highlighting information to the GUI. This can be useful (faster) if highlighting multiple groups of nets in a loop, since each highlight command that affects a net will otherwise refresh the entire routing data set in the GUI.
`[-clear]`	Y	The optional -clear option is used to clear all prior highlighting

ignore_cancel

`ignore_cancel <script>`

Temporarily ignore cancel button. Useful to execute cleanup commands in a flow step after a cancel has been caught.

Argument	Optional	Description
`<script>`		commands to execute

initialize_flow

`initialize_flow`

This command clears the current flow model, then sources the master flow.tcl script. The master flow.tcl script uses these flow TCL commands to define the default flow.

insert_delay

`insert_delay <pinlist>`

This command parses the user directive to add extra delays for paths that require additional delay for alleviating timing violations.

Argument	Optional	Description
`<pinlist>`		The required {pinlist} option is used to specify the load pins of a net that need to be driven by inserted gates. For each pin, you can optionally specify an integer delay value by using the format <delay>,<pin_name>. The default delay value is 1.

is_incremental_compile

`is_incremental_compile`

Check if the Incremental Compile Impl Option is set to 1, and that partitions have been defined on the design via the *.prt file

is_labmode

`is_labmode`

This command returns 1 if we are in labmode .

load_flowscripts

`load_flowscripts [-bitstream]`

This command loads all of the encrypted flow scripts.

Argument	Optional	Description
`[-bitstream]`	Y	load only scripts relevant for bitstream

load_project

`load_project <projectFile> [-not_active] [-force]`

This command loads a project file into ACE. Loading a project file does not load the design files, it just sets up a project for later use.

Argument	Optional	Description
`<projectFile>`		The required <projectFile> argument specifies the path to a project file.
`[-not_active]`	Y	If this option is set, no impl in the project will be activated and the active impl in ACE will not be changed.
`[-force]`	Y	The -force option can be used to override a project lock that has been set by another ACE session. Using -force causes the current ACE session to take ownership of the project lock for the project being restored. DO NOT use this option to run multiple ACE sessions on the same project at the same time, or else output files (acxprj, acxdb, icdb, jam, etc) and log files may become corrupted!

message

```
message <msg> [-info] [-warning] [-error] [-console_off] [-console_on]
```

This command prints a status message to the console and the log file.

Argument	Optional	Description
`<msg>`		The message to be printed.
`[-info]`	Y	Make this message an informational message.
`[-warning]`	Y	Make this message a warning message.
`[-error]`	Y	Make this message an error message.
`[-console_off]`	Y	turn off console io
`[-console_on]`	Y	turn on console io

move_project_constraints

```
move_project_constraints [-project <string>] <file> <offset>
```

This command moves a project constraints file to the specified offset to allow re-ordering of constraints within a project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to move constraints for.
`<file>`		The project constraints file to move.
`<offset>`		The offset to move the project constraints file to. Other constraints files will be moved down automatically.

move_project_netlists

`move_project_netlists [-project <string>] <file> <offset>`

This command moves a project netlist file to the specified offset to allow re-ordering of netlists within a project.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) to move netlists for.
`<file>`		The project netlist file to move.
`<offset>`		The offset to move the project netlist file to. Other netlist files will be moved down automatically.

move_relative_paths

`move_relative_paths <line> <source> <target>`

Change relative paths in input string; starting out relative to parameter source ,returned as relative to parameter target.

Argument	Optional	Description
`<line>`		line containing a quoted string of paths.
`<source>`		The location of the original script containing <line>
`<target>`		The script that will contain <line> and still work.

optimize_tile

`optimize_tile [-verbose] [-timing <int>] [-optimize_fixed]`

optimize placement of single tile or tiles seperately

Argument	Optional	Description
`[-verbose]`	Y	use verbose option reporting
`[-timing <int>]`	Y	timing-mode: 1 = timing-driven
`[-optimize_fixed]`	Y	change placement of fixed instances - use with caution !

redirect

`redirect <command> [-variable <string>]`

redirect messages to variable.

Argument	Optional	Description
`<command>`		command to run

Argument	Optional	Description
`[-variable <string>]`	Y	name of variable

refresh_drawing

```
refresh_drawing
```

This command refreshes the current custom drawing on the GUI's Floorplanner view.

regenerate_all_ip_design_files

```
regenerate_all_ip_design_files [-ioringOutputDir <string>] [-add_to_project]
```

This command re-generates design files the all ACXIP files in the active ACE project, for both Core fabric IP and IO Ring design.

Argument	Optional	Description
`[-ioringOutputDir <string>]`	Y	The -ioringOutputDir option allows the user to specify a non-default directory path to output the IO Ring design files into.
`[-add_to_project]`	Y	Using the -add_to_project option automatically adds the required generated IO Ring design files to your ACE project. This includes utilization XML, SDC constraints, PDC constraints, and IO Ring bitstream files.

remap_partial_bitstream

```
remap_partial_bitstream <hex_file> <original_clusters> <new_clusters> [-output_file <string>]
```

This command returns a list of compatible ordering codes for the active project based on it's device, package, and speed grade.

Argument	Optional	Description
`<hex_file>`		The path to the bitstream hex file.
`<original_clusters>`		Provide a list of clusters from your original design (i.e. {{LOGIC_CLUSTER_1_1} {LOGIC_CLUSTER_1_2} {LOGIC_CLUSTER_2_1} {LOGIC_CLUSTER_2_2}}).
`<new_clusters>`		Provide a list of clusters from your partially reconfigured design (i.e. {{LOGIC_CLUSTER_6_4} {LOGIC_CLUSTER_6_5} {LOGIC_CLUSTER_7_4} {LOGIC_CLUSTER_7_5}}).
`[-output_file <string>]`	Y	Optional output file. If no output file is specified, the command outputs the file to <original_hex_file>_remapped.hex

remove_clock_preroute

```
remove_clock_preroute <net_name> <track_list> [-clock_regions <list>] [-clusters <list>] [-placement_regions <list>] [-partitions <list>]
```

This command will remove the pre-routing constraints from a clock or reset net on the clock tracks and regions specified.

Argument	Optional	Description
`<net_name>`		The name of the clock or reset net to remove from pre-routing.
`<track_list>`		The list of integer clock track numbers to remove from pre-route on this net. Valid clock track numbers are device-specific.
`[-clock_regions <list>]`	Y	The list of clock region names to remove from pre-route on this net. Valid clock region names are device-specific.
`[-clusters <list>]`	Y	The list of cluster names to remove from pre-route on this net. Valid cluster names are device-specific.
`[-placement_regions <list>]`	Y	The list of placement region names to remove from pre-route on this net. Valid placement region names are device-specific.
`[-partitions <list>]`	Y	The list of partition names to remove from pre-route on this net. Valid partition are device-specific.

remove_flow_step

```
remove_flow_step <id>
```

This command removes an existing flow step from ACE only if the flow step is a user-defined flow step.

Argument	Optional	Description
`<id>`		The required <id> argument specifies the id of the flow step to remove.

remove_impl

```
remove_impl <implNames_list> [-project <string>]
```

This command removes multiple implementations from a project. The implementations' output directories on the file system are not deleted.

Argument	Optional	Description
`<implNames_list>`		The required <implNames_list> argument is used to specify the names of the implementations to remove.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the implementation to be removed from.

remove_path

```
remove_path [<id>] [-all]
```

This command removes a user-defined pin path.

Argument	Optional	Description
[<id>]	Y	Specifies the id of the path to remove
[-all]	Y	Removes all paths

remove_project

`remove_project <projectName>`

This command removes a project from ACE. The project file on disk is not deleted.

Argument	Optional	Description
<projectName>		The required <projectName> argument is used to specify the project to be removed (by name).

remove_project_constraints

`remove_project_constraints <files> [-project <string>]`

This command removes the link to an SDC, PDC, or TCL constraint file from a project. The SDC constraint file on disk is not deleted.

Argument	Optional	Description
<files>		The required <files> argument is used to specify the SDC, PDC, or TCL constraint files (by file path).
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the SDC constraint file to be removed from.

remove_project_constraints_pvt

`remove_project_constraints_pvt <file>`

This command allows the user to remove all PVT conditions from an SDC constraint file.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the SDC constraint file.

remove_project_ip

`remove_project_ip <list_of_files> [-project <string>]`

This command removes the association from a project to one or more IP settings files. The IP settings files on disk are not deleted.

Argument	Optional	Description
`<list_of_files>`		The required <list_of_files> argument is used to specify the IP settings files (by file path). The file paths may be absolute, or may be relative to the acxprj file's directory.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be removed from. The named project must already be opened in ACE.

remove_project_netlist

```
remove_project_netlist <files> [-project <string>] [-impl <string>]
```

This command removes the link to a verilog netlist file from a project. The verilog netlist file on disk is not deleted.

Argument	Optional	Description
`<files>`		The required <file> argument is used to specify the verilog netlists (by file path).
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be removed from.
`[-impl <string>]`	Y	If a file belongs to an impl and should be removed only from that impl, specify that impl here.

remove_region

```
remove_region [-region <string>] [-all]
```

This command removes a placement region constraint specification

Argument	Optional	Description
`[-region <string>]`	Y	Name of the region to delete
`[-all]`	Y	Remove all region constraints

remove_region_insts

```
remove_region_insts <region> [-insts <list>] [-all] [-flops_only] [-clocks_only] [-
verbose]
```

Remove user design instances from an existing placement region constraint

Argument	Optional	Description
`<region>`		name of the region to clear
`[-insts <list>]`	Y	List of user design instances to remove from this placement region constraint.
`[-all]`	Y	Clear all user design instances from this region constraint

Argument	Optional	Description
[-flops_only]	Y	When removing instances, filter out all instances except flops
[-clocks_only]	Y	When removing instances, filter out all instances with no connected clock
[-verbose]	Y	Print additional debug messages.

rename_impl

`rename_impl <newImplName> [-project <string>] [-impl <string>]`

This command renames an implementation. Changing the name of an implementation also changes the name of the implementation output directory on disk (even without calling "save_project").

Argument	Optional	Description
<newImplName>		The required <newImplName> argument is used to specify the new implementation name.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to change the name for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to change the name for.

report_clock_regions

`report_clock_regions [-outputfile <string>] [-text] [-csv]`

This command generates and writes a formatted report showing which clock nets are routed in each clock region

Argument	Optional	Description
[-outputfile <string>]	Y	The -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation debug directory and is named <design_name>_regions.html.
[-text]	Y	The -text option is used to specify whether the file should be output to the console as plain text
[-csv]	Y	The -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets

report_clocks

`report_clocks`

Report clocks in the current design

report_coverage

`report_coverage [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>] [-verbose]`

Generate and write a coverage report for pins.

Argument	Optional	Description
[-outputfile <list>]	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <design_name>_pins.html.
[-text]	Y	The optional -text option is used to specify whether the file should be output as plain text.
[-csv]	Y	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	Y	The optional -html option is used to specify whether the file should be output as html-file.
[-columns <list>]	Y	The optional -columns option is used to specify a customized ordered list of columns names to output in the pin assignment report.
[-verbose]	Y	The optional -verbose option is used to specify whether the file should report ALL attributes and parameters for each IO port/pad.

report_design_stats

```
report_design_stats [-outputfile <list>] [-html] [-csv] [-text]
```

This command generates and writes a formatted report about various design statistics

Argument	Optional	Description
[-outputfile <list>]	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_design_stats, with the extension .html or .csv (respectively).
[-html]	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	Y	The -text option specifies that the output file(s) are written in plain text format

report_impl_options

```
report_impl_options [-outputfile <string>] [-text] [-csv] [-project <string>] [-impl
<string>] [-hide_values] [-show_standard] [-diff_options]
```

Output a report of the current impl options defined in ACE. If no -project and/or -impl options are specified, the active impl will be reported.

Argument	Optional	Description
[-outputfile <string>]	Y	The optional -outputfile option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <design_name>_impl_options.html.
[-text]	Y	The optional -text option is used to specify that the file should be output as plain text.
[-csv]	Y	The optional -csv option is used to specify that the file should be output as a CSV file for use in Excel spreadsheets.
[-project <string>]	Y	The optional -project <projectName> option must be used with -impl <implName>. These options are used to specify an alternate project implementation (by name).
[-impl <string>]	Y	The optional -impl <implName> option can be used with or without the -project <projectName> option to specify an impl. Without -project <projectName>, -impl <implName> finds the impl in the active project.
[-hide_values]	Y	If the -hide_values flag is used, the report will NOT output an additional column to display the current values for the given impl.
[-show_standard]	Y	If the -show_standard flag is used, the report and optional diff options Tcl file will ONLY output standard options that show in the GUI. If this flag is not set, by default, all available options for the active impl will be output.
[-diff_options]	Y	The optional -diff_options flag is used to create a Tcl file containing the full set of implementation options with values which vary from the default values. Each impl option's default value is listed as a comment.

report_partitions

`report_partitions [-outputfile <list>] [-html] [-csv] [-text]`

This commands generates and writes a formatted partition report

Argument	Optional	Description
[-outputfile <list>]	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_partitions, with the extension .html or .csv (respectively).
[-html]	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	Y	The -text option specifies that the output file(s) are written in plain text format

report_performance

`report_performance [-outputfile <list>] [-html] [-csv] [-text] [<num_paths>] [-setup_pass_only]`

Generate a report detailing the estimated design performance and the quality of the mapping in ACE, including a critical path breakdown

Argument	Optional	Description
`[-outputfile <list>]`	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_performance_report, with the extension .html or .csv (respectively).
`[-html]`	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
`[-csv]`	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
`[-text]`	Y	The -text option specifies that the output file(s) are written in plain text format
`[<num_paths>]`	Y	The number of critical paths to analyze in each clock domain (default is 3)
`[-setup_pass_only]`	Y	If set, only paths that meet timing are logged

report_pins

`report_pins [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>]`

Generate and write a pin to package assignment report

Argument	Optional	Description
`[-outputfile <list>]`	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <design_name>_pins.html.
`[-text]`	Y	The optional -text option is used to specify whether the file should be output as plain text.
`[-csv]`	Y	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
`[-html]`	Y	The optional -html option is used to specify whether the file should be output as html-file.

Argument	Optional	Description
[-columns <list>]	Y	The optional -columns option is used to specify a customized ordered list of columns names to output in the pin assignment report.

report_placement

`report_placement [-outputfile <list>] [-html] [-csv] [-text]`

This command generates and writes a formatted placement QoR report

Argument	Optional	Description
[-outputfile <list>]	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_placement, with the extension .html or .csv (respectively).
[-html]	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	Y	The -text option specifies that the output file(s) are written in plain text format

report_power

`report_power [-outputfile <list>] [-html] [-csv] [-text] [-temperature <string>] [-clocks <string>] [-achieved] [-saif_file <string>] [-saif_top_level <string>]`

This command generates and writes a formatted power dissipation report

Argument	Optional	Description
[-outputfile <list>]	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_power, with the extension .html or .csv (respectively).
[-html]	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	Y	The -text option specifies that the output file(s) are written in plain text format
[-temperature <string>]	Y	Override the junction temperature printed in the report header

Argument	Optional	Description
[-clocks <string>]	Y	The -clocks <{ {clk1 freq} {clk2 freq} .. {clkn freq} }> option may be used to specify the operating frequency (in MHz) of all the clocks in the design. If this option is not present, the frequencies from the design constraints will be used by default. If no constraints are found, the best achieved frequency will be used.
[-achieved]	Y	The -achieved option may be used to specify that achieved static timing results be used to calculate the power dissipation report for each clock
[-saif_file <string>]	Y	The -saif_file option may be used to specify the path to a .saif file. Switching Activity Interchange Format (saif) files can be used to provide a more accurate power profile of a design.
[-saif_top_level <string>]	Y	The -saif_top_level option may be used to specify the name of the top level instance in the saif file. Default name used is "DUT".

report_routing

```
report_routing [-outputfile <list>] [-text] [-csv] [-html] [-nonterse] [-terse] [-wl] [-
overflowreportlimit <int>]
```

This command generates and writes a formatted routing report.

Argument	Optional	Description
[-outputfile <list>]	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation debug directory and is named <design_name>_routing.html.
[-text]	Y	The optional -text option is used to specify whether the file should be output as plain text (Default is autodetect)
[-csv]	Y	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	Y	The optional -html option is used to specify whether the file should be output as html-file.
[-nonterse]	Y	normal information level.
[-terse]	Y	terse info level.
[-wl]	Y	wire length report.
[-overflowreportlimit <int>]	Y	limit on the number of overflows. Defaults to 11.

report_utilization

`report_utilization [-outputfile <list>] [-html] [-csv] [-text]`

This command generates and writes a formatted device utilization report

Argument	Optional	Description
`[-outputfile <list>]`	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_utilization, with the extension .html or .csv (respectively).
`[-html]`	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
`[-csv]`	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
`[-text]`	Y	The -text option specifies that the output file(s) are written in plain text format

reset_impl_option

`reset_impl_option [<option_name>] [-all] [-project <string>] [-impl <string>]`

This command resets a project implementation option to its (device-specific) default value. Only one option may be reset at a time.

Argument	Optional	Description
`[<option_name>]`	Y	The name of the impl option to reset to its default. To see a list of valid impl options, use the report_impl_options TCL command.
`[-all]`	Y	The optional -all option resets all impl options to their device-specific default values.
`[-project <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.
`[-impl <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.

restore_impl

`restore_impl <filename> [-project <string>] [-impl <string>]`

The restore_impl command loads an ACXDB file to restore the state of the DB and impl options for a given impl. Restoring an impl automatically makes that impl the active impl. If no -impl and -project options are specified, then the ACXDB file is loaded for the current active impl. By default, the DB will be restored using all saved information in the ACXDB file, including placement and routing information. Restoring an impl overrides the current impl option values with the impl option values saved in the ACXDB file. Restoring an impl clears the current state of the DB for the current active impl, so be sure to save your active impl before restoring an impl.

Argument	Optional	Description
`<filename>`		Specifies the ACXDB file path to restore the state of the active impl from
`[-project <string>]`	Y	Specifies an alternate project name to use instead of the active project when using the -impl <implname> option
`[-impl <string>]`	Y	Specifies an alternate impl name to restore instead of restoring the current active impl

This functionality is also accessible through buttons/menus in the ACE GUI as described in Restoring Implementations (see page 277).

> **Restoring an Implementation clears current data**
>
> Restoring an Implementation (see page 215) will first clear all data in memory before beginning the restore process. Any data that has not been saved will be lost.
>
> The restored implementation (and project) will become the Active Project and Active Implementation, and all Implementation Options will also be restored from file, overwriting any values currently in memory.

See also: `save_impl` and Saving Implementations (see page 276).

restore_project

```
restore_project <projectFile> [-reload] [-not_active] [-no_db] [-activeimpl <string>] [-
acxdb <string>] [-force]
```

The restore_project command loads an ACE project (.acxprj) file and restores the project's implementation options. The -acxdb option can be used to specify the file path of the ACXDB file to restore the DB state for the active implementation. The -activeimpl option can be used to specify the impl name to activate and restore. If -not_active is used, no impl in the project will be activated or restored from its ACXDB file and the active impl in ACE will not be changed.

Argument	Optional	Description
`<projectFile>`		Specifies the ACE project (.acxprj) file path to load and restore.
`[-reload]`	Y	Use this option to re-load the ACE project (.acxprj) file from disk. This will clear the design DB and flow status, requiring the design to be re-run from the beginning of the flow.
`[-not_active]`	Y	If this option is set, no impl in the project will be activated or restored from its ACXDB file and the active impl in ACE will not be changed.
`[-no_db]`	Y	If this option is set, the DB state of the active impl for the project will not be restored.
`[-activeimpl <string>]`	Y	The -activeimpl option can be used to specify an alternate impl name to activate and restore. By default, the last active impl during the session the project was saved in will be activated.
`[-acxdb <string>]`	Y	Specifies an ACXDB file path from which to restore the state of the active impl.
		The -force option can be used to override a project lock that has been set by another ACE session. Using -force causes the current ACE session to take ownership of the project lock for

Argument	Optional	Description
`[-force]`	Y	the project being restored. DO NOT use this option to run multiple ACE sessions on the same project at the same time, or else output files (acxprj, acxdb, icdb, jam, etc) and log files may become corrupted!

run

`run [-step <string>] [-stop_at_step <string>] [-resume] [-ic <string>]`

This command runs the steps of the design flow. It can be used to run the entire flow from the beginning, run a specific flow step, or resume the flow from the last incomplete step. Using no options will run the entire flow from the beginning. The default Achronix flow step IDs (for those options requiring them) are: {prepare run_prepare report_timing_prepared write_netlist_prepared place_and_route run_place report_timing_placed run_route report_timing_routed design_completion post_process final_drc_checks report_timing_final write_netlist_final fpga_program write_bitstream fpga_download} Because advanced users may create their own flow steps, a complete list of all flow step IDs can be retrieved with the Tcl command 'get_flow_steps'.

Argument	Optional	Description
`[-step <string>]`	Y	The optional -step <id> option is used to run the specified flow step, by ID, (along with any incomplete required pre-requisite steps,) and all of its children. See 'get_flow_steps' for a list of all IDs.
`[-stop_at_step <string>]`	Y	The optional -stop_at_step <id> option is used to stop the flow after running the specified flow step, by ID. See 'get_flow_steps' for a list of all IDs.
`[-resume]`	Y	The optional -resume option is used to run the entire flow from the last successfully completed flow step.
`[-ic <string>]`	Y	The optional -ic init|continue option specifies incremental compilation flow modes. 'init' implies beginning of the flow without using previous state of compiled design and 'continue' implies incrementally compiling of previous state of design.

run_fanout_control

`run_fanout_control [-physical <int>] [-fanout_limit <int>] [-fanout_limit_clone <int>]`

This command does fanout control for high fanout control nets

Argument	Optional	Description
`[-physical <int>]`	Y	Clone Critical Instances that have slack lower than this limit
`[-fanout_limit <int>]`	Y	Apply fanout control on nets with fanout greater than this limit
`[-fanout_limit_clone <int>]`	Y	Apply fanout cloning on nets with fanout greater than this limit

run_final_drc_checks

`run_final_drc_checks`

This command performs final DRC checks on the active design. If there is currently no active project/implementation, the reportsdir and debugdir must be specified.

run_fpga_download

`run_fpga_download [-outputdir <string>] [-download_pod_names <string>] [-jam_file <string>]`

This command downloads the generated bitstream to the target device.

Argument	Optional	Description
`[-outputdir <string>]`	Y	Output directory name
`[-download_pod_names <string>]`	Y	(Optional) JTAG programming device name. If this is not specified, auto-detection of JTAG programming devices will be attempted, and connection will fail if more than one JTAG programming device is auto-detected.
`[-jam_file <string>]`	Y	Optional jam (STAPL) file to download. The default jam file in your output directory will be used if this is not specified

run_generate_bitstream

`run_generate_bitstream [-outputdir <string>] [-aeskey <string>]`

This command generates a bitstream file for programming the target device.

Argument	Optional	Description
`[-outputdir <string>]`	Y	Output directory name
`[-aeskey <string>]`	Y	Key used for encryption. If not given, key is taken from impl. If not active in impl, the bitstream is not encrypted.

run_generate_final_reports

`run_generate_final_reports [-name_postfix <string>] [-format <string>]`

This command generates various report files, including clocks, pins, power, etc. Implementation options are used to control which report files are generated.

Argument	Optional	Description
`[-name_postfix <string>]`	Y	Postfix added to report file name (e.g., to distinguish multiple 'placed' reports)
`[-format <string>]`	Y	Specify report formats; default is { text html csv }

run_generate_fullchip_sim

`run_generate_fullchip_sim [-debugdir <string>] [-modelsdir <string>]`

This command generates the files necessary for fullchip simulation.

Argument	Optional	Description
`[-debugdir <string>]`	Y	The -debugdir <dir> option is used to override the default location for debug files during this step.
`[-modelsdir <string>]`	Y	The -modelsdir <dir> option is used to override the default location for the fullchip sim top-level models.

run_generate_netlist

`run_generate_netlist [-outputfile <string>] [-final] [-compress]`

This command generates a verilog netlist for simulation.

Argument	Optional	Description
`[-outputfile <string>]`	Y	Output netlist file name.
`[-final]`	Y	Output DRC-free final netlist
`[-compress]`	Y	Compress output file with gzip

run_insert_holdbuffers

`run_insert_holdbuffers [-margin <int>] [-io_buffers <int>] [-typebased_buffers <int>]`

This command generates extra gate delays by inserting a buffer per target pin if that pin has a hold time slack value that is less than the margin specified.

Argument	Optional	Description
`[-margin <int>]`	Y	Insert a delay buffer per target pin whose hold time slack is less than the specified value
`[-io_buffers <int>]`	Y	Insert a delay buffer per target pin when driven directly by a flopped IO Pad/Pin
`[-typebased_buffers <int>]`	Y	Insert a delay buffer per target pin according to the given cell-type bitfield specification (2^0 = DFF inputs, 2^1 = DFF outputs, 2^2 = clocked OPIN inputs, 2^3 = clocked IPIN outputs, 2^4=BRAM inputs, 2^5 = BRAM outputs, 2^6 = LRAM inputs, 2^7 = LRAM outputs, 2^8 = DSP inputs, 2^9 = DSP outputs, 2^10 = MLP inputs, 2^11 = MLP outputs, 2^12 = NAP inputs, 2^13 = NAP outputs). All other values are reserved for future use and should be unset.)

run_multiprocess

```
run_multiprocess [-use_existing_impls <list>] [-use_seeds <list>] [-parallel_job_count
<int>] [-use_job_submission <int>] [-stop_flow_at <string>] [-copy_icdb <int>] [-
jobs_exec <string>] [-jobs_wd <string>] [-jobs_name <string>] [-jobs_log <string>] [-
jobs_args <list>] [-jobs_nfs_latency <int>] [-create_option_sets] [-remove_nonbest]
```

This command runs the ACE multiprocess flow for the active implementation. To generate new implementations from option sets and run multiprocess, use -create_option_sets. NOTE: For any optional arguments that are not specified, the current Multiprocess configuration from the ACE GUI User Preferences will be used as defaults.

Argument	Optional	Description
`[-use_existing_impls <list>]`	Y	The use_existing_impls option allows the user to specify a list of existing impl names to run in multiprocess, instead of using seed sweep or generating impls from option sets. To run all existing impls, you can specify -use_existing_impls [ace::get_impl_names]
`[-use_seeds <list>]`	Y	The use_seeds option allows the user to specify a list of PnR seed values to run in multiprocess, instead of using existing impls or generating impls from option sets.
`[-parallel_job_count <int>]`	Y	(optional) Sets the number of implementations to run in parallel. If not specified, defaults to the GUI's preference setting.
`[-use_job_submission <int>]`	Y	(optional) Set to a 0 to run background jobs on the local machine, or set to a 1 to submit jobs to a cloud/grid/batch submission system. If not specified, defaults to the GUI's preference setting.
`[-stop_flow_at <string>]`	Y	(optional) If a valid flow step ID is specified, that flow step will be enabled and will be the last flow step executed by all implementations. If not specified, ACE will run the entire flow (ignores user's GUI preference setting).
`[-copy_icdb <int>]`	Y	(optional) Set to a 1 to copy the incremental flow DB from the template impl, or set to a 0 to not copy. Defaults to 0 (no file copy) if not specified (ignores user's GUI preference setting).
`[-jobs_exec <string>]`	Y	(optional) Specify the job submission system executable. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_wd <string>]`	Y	(optional) Specify the job submission system working directory argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_name <string>]`	Y	(optional) Specify the job submission system job name argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_log <string>]`	Y	(optional) Specify the job submission system job log argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_args <list>]`	Y	(optional) Specify the job submission system list of additional arguments, each with at most one optional value, formatted as a list of TCL lists in the form: {{arg1 val1} {arg2} {arg3 val3} ... }. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
		(optional) Specify the allowed seconds of NFS write latency (how long to wait between process completion and reading the files generated by the just-finished

Argument	Optional	Description
[-jobs_nfs_latency <int>]	Y	process). This is relevant both to local background jobs, and job submission systems. If not specified, defaults to the GUI's preference setting.
[-create_option_sets]	Y	(optional) Auto-generate option sets relevant to the active implementation, which will appear the active impl's option_sets directory. Note that this is only relevant when neither -use_existing_impls nor -use_seeds are active (meaning we're in the default mode, generating new impls for option sets).
[-remove_nonbest]	Y	(optional) Removes all recently generated implementations from the Projects View, except for the base impl and best impl, at the end of the Multiprocess run.

Default Values

For any parameters that are not specified, the ACE GUI user preferences will be used.

For example, if `-parallel_job_count` is not explicitly specified, and if your ACE GUI user preferences are currently configured to use four (4) jobs, that value will be used for your batch multiprocess run.

A more detailed description of the use of `run_multiprocess` can be found in the Multiprocess Batch Mode (see page 292) section.

The GUI provides a graphical interface for multiprocess through the Multiprocess View (see page 83). See also: Running Multiple Flows in Parallel (see page 282), Attempting Likely Optimizations Using Option Sets (see page 352).

run_multiprocess_iterator

```
run_multiprocess_iterator [-use_existing_impls <list>] [-use_seeds <list>] [-
parallel_job_count <int>] [-use_job_submission <int>] [-stop_flow_at <string>] [-
copy_icdb <int>] [-jobs_exec <string>] [-jobs_wd <string>] [-jobs_name <string>] [-
jobs_log <string>] [-jobs_args <list>] [-jobs_nfs_latency <int>] [-create_option_sets] [-
remove_nonbest] [-iterations <int>]
```

This command runs the ACE multiprocess flow for the active implementation for 5 (default) iterations, each time selecting the 'best' impl option for the subsequent multiprocess run. To run the same multiprocess flow as run_multiprocess, this command is exposed to all options from run_multiprocess. To run multiprocess with option sets, use -create_option_sets.

Argument	Optional	Description
[-use_existing_impls <list>]	Y	The use_existing_impls option allows the user to specify a list of existing impl names to run in multiprocess, instead of using seed sweep or generating impls from option sets. To run all existing impls, you can specify -use_existing_impls [ace::get_impl_names]
[-use_seeds <list>]	Y	The use_seeds option allows the user to specify a list of PnR seed values to run in multiprocess, instead of using existing impls or generating impls from option sets.
[-parallel_job_count <int>]	Y	(optional) Sets the number of implementations to run in parallel. If not specified, defaults to the GUI's preference setting.

Argument	Optional	Description
`[-use_job_submission <int>]`	Y	(optional) Set to a 0 to run background jobs on the local machine, or set to a 1 to submit jobs to a cloud/grid/batch submission system. If not specified, defaults to the GUI's preference setting.
`[-stop_flow_at <string>]`	Y	(optional) If a valid flow step ID is specified, that flow step will be enabled and will be the last flow step executed by all implementations. If not specified, ACE will run the entire flow (ignores user's GUI preference setting).
`[-copy_icdb <int>]`	Y	(optional) Set to a 1 to copy the incremental flow DB from the template impl, or set to a 0 to not copy. Defaults to 0 (no file copy) if not specified (ignores user's GUI preference setting).
`[-jobs_exec <string>]`	Y	(optional) Specify the job submission system executable. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_wd <string>]`	Y	(optional) Specify the job submission system working directory argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_name <string>]`	Y	(optional) Specify the job submission system job name argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_log <string>]`	Y	(optional) Specify the job submission system job log argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_args <list>]`	Y	(optional) Specify the job submission system list of additional arguments, each with at most one optional value, formatted as a list of TCL lists in the form: {{arg1 val1} {arg2} {arg3 val3} ... }. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
`[-jobs_nfs_latency <int>]`	Y	(optional) Specify the allowed seconds of NFS write latency (how long to wait between process completion and reading the files generated by the just-finished process). This is relevant both to local background jobs, and job submission systems. If not specified, defaults to the GUI's preference setting.
`[-create_option_sets]`	Y	(optional) Auto-generate option sets relevant to the active implementation, which will appear the active impl's option_sets directory. Note that this is only relevant when neither -use_existing_impls nor -use_seeds are active (meaning we're in the default mode, generating new impls for option sets).
`[-remove_nonbest]`	Y	(optional) Removes all recently generated implementations from the Projects View, except for the base impl and best impl, at the end of the Multiprocess run.
`[-iterations <int>]`	Y	The iterations option allows the user to specify the number of iterations between 1 and 5. Default is 5.

Warning: Early Access Functionality

This multiprocess iterator (along with the QoR sorting of the Multiprocess Summary Report (see page 238)) should currently be considered early-access functionality. Future ACE releases may improve QoR, reduce runtimes, and require fewer iterations to achieve similar results.

See also: `run_multiprocess`, Multiprocess View (see page 83), Running Multiple Flows in Parallel (see page 282), Attempting Likely Optimizations Using Option Sets (see page 352), Multiprocess Batch Mode (see page 292)

run_place

```
run_place
```

This command clears all routing and places the design.

run_post_process

```
run_post_process
```

This command post-processes the routed design to insert reset and other Achronix-specific technologies.

run_prepare

```
run_prepare [-ic <string>]
```

This command clears the current netlist and constraints data, then loads all the design files for the active implementation, runs design checks, and compiles the design into an Achronix design.

Argument	Optional	Description
`[-ic <string>]`	Y	The optional -ic init|continue option specifies incremental compilation flow modes. 'init' implies beginning the flow without using previous state of compiled design and 'continue' implies incrementally compiling of previous state of design.

run_route

```
run_route
```

This command routes the design.

run_secureshare

```
run_secureshare [-read_manifest <string>] [-generate_manifest <string>] [-archive_path <string>] [-encrypt] [-wizard] [-force]
```

Read or generate a SecureShare Manifest File, and create a zipped (and optionally encrypted) archive containing project source, database, log, report, and debug files. To receive better support, please attach this zip to your Achronix support ticket.

Argument	Optional	Description
`[-read_manifest <string>]`	Y	Manifest file path read to generate an archive (unless -no_archive is specified). Cannot be used with -generate_manifest.

Argument	Optional	Description
[-generate_manifest <string>]	Y	Manifest file output path, and default archive output path (unless -archive_path is specified). Cannot be used with -read_manifest.
[-archive_path <string>]	Y	Archive output path, which can override the output directory within a manifest when using -read_manifest
[-encrypt]	Y	Encrypt the archive. With -read_manifest, the archive is always encrypted (unless -no_archive is specified). With -generate_manifest, the manifest is marked for encryption, and the generated archive is encrypted (unless -no_archive is specified).
[-wizard]	Y	Generate a manifest and then open the GUI file-picker.
[-force]	Y	Overwrite generated output files.

Example

Running the command with no arguments will automatically generate a SecureShare manifest and archive using the current active impl:

```
run_secureshare
```

If you'd like to create the archive in a specific location, rather than the project default, use:

```
run_secureshare -archive_path <output_path>
```

> The file extension for a regular SecureShare archive will always be .zip

To encrypt the archive, use:

```
run_secureshare -encrypt
```

> The file extension for an encrypted SecureShare archive will always be .zip.encrypted

To generate an archive from a previously created manifest, use:

```
run_secureshare -read_manifest <manifest_path.acxssm>
```

This will read the .acxssm manifest file, which includes the archive output directory and encryption status, and generate an archive using these options.
You can override the manifest with the -archive_path <output_path> and -encrypt options.

If you'd like to automatically create a manifest without generating an archive, use:

```
run_secureshare -no_archive
```

To specify the manifest path, use:

```
run_secureshare -generate_manifest <manifest_path>
```

The file extension for a SecureShare manifest will always be .acxssm

To overwrite manifest and archive outputs, as well as create missing directories, use:

```
run_secureshare -force
```

run_snapshot

`run_snapshot <snapshotFile> [-pod_name <string>] [-ir_bits_before <int>] [-ir_bits_after <int>] [-target_offset <int>] [-timeout <int>] [-verbose]`

This command runs the Snapshot Debugger for a given Snapshot configuration (.snapshot file). This command outputs a VCD file and a Log file. The file paths are specified in the Snapshot configuration file.

Argument	Optional	Description
`<snapshotFile>`		The required <snapshotFile> argument specifies the Snapshot configuration (.snapshot file) to be used by Snapshot debugger.
`[-pod_name <string>]`	Y	(optional) specifies which JTAG pod connection to use. (If not specified, the current JTAG Connection configuration from the GUI User Preferences will be the default.)
`[-ir_bits_before <int>]`	Y	(optional) Sets the (decimal) number of instruction register bits between the board JTAG TDI pin and the target device. Use 0 for single-device JTAG scan chains. (If not specified, the current JTAG Connection configuration from the GUI User Preferences will be the default.)
`[-ir_bits_after <int>]`	Y	(optional) Sets the (decimal) number of instruction register bits between the target device and the board JTAG TDO pin. Use 0 for single-device JTAG scan chains. (If not specified, the current JTAG Connection configuration from the GUI User Preferences will be the default.)
`[-target_offset <int>]`	Y	(optional) Sets the device count (in decimal) between the board JTAG TDI pin and target FPGA device. Use 0 for single-device JTAG scan chains. (If not specified, the current JTAG Connection configuration from the GUI User Preferences will be the default.)
`[-timeout <int>]`	Y	(optional) specifies the timeout in seconds before Snapshot is cancelled. If not specified, Snapshot will not timeout.
`[-verbose]`	Y	(optional) Exposes additional log info when running Snapshot.

run_stapl_action

`run_stapl_action <stapl_file> <action> [-pod_name <string>] [-ir_bits_before <int>] [-ir_bits_after <int>] [-target_offset <int>] [-disabled_procs <list>] [-enabled_procs <list>] [-defines <list>] [-log_file <string>]`

This command executes the given stapl program action.

Argument	Optional	Description
`<stapl_file>`		(required) specifies which STAPL file will contain the Action to be run.
`<action>`		(required) specifies which STAPL Action will be run.
`[-pod_name <string>]`	Y	(optional) specifies which JTAG pod connection to use. (If not specified, autodetection will be attempted.)
`[-ir_bits_before <int>]`	Y	(optional) Sets the (decimal) number of instruction register bits between the board JTAG TDI pin and the target device. Use 0 for single-device JTAG scan chains. (If not specified, the value embedded within the STAPL will be used.)
`[-ir_bits_after <int>]`	Y	(optional) Sets the (decimal) number of instruction register bits between the target device and the board JTAG TDO pin. Use 0 for single-device JTAG scan chains. (If not specified, the value embedded within the STAPL will be used.)
`[-target_offset <int>]`	Y	(optional) Sets the device count (in decimal) between the board JTAG TDI pin and target FPGA device. Use 0 for single-device JTAG scan chains. (If not specified, the value embedded within the STAPL will be used.)
`[-disabled_procs <list>]`	Y	(optional) specifies which recommended STAPL Procedures in the specified Action should be skipped.
`[-enabled_procs <list>]`	Y	(optional) specifies which optional STAPL Procedures in the specified Action should be executed.
`[-defines <list>]`	Y	(optional) list of definitions to add (e.g. 'rw_addr_int=4100').
`[-log_file <string>]`	Y	(optional) specifies the path to the log file which will be populated with the STAPL Player's console output. (NOTE: If file exists, it will be overwritten.)

run_timing_analysis

```
run_timing_analysis [-prepared] [-placed] [-routed] [-final] [-name_postfix <string>] [-format <string>] [-temperature <string>] [-voltage <string>] [-grade <string>]
```

This command runs timing analysis on the design.

Argument	Optional	Description
`[-prepared]`	Y	Indicates that the design has only been prepared (this is the default)
`[-placed]`	Y	Indicates that the design has been placed but not routed
`[-routed]`	Y	Indicates that the design has been placed and routed

Argument	Optional	Description
`[-final]`	Y	Indicates that this is sign-off timing (this involves some extra checks)
`[-name_postfix <string>]`	Y	Postfix added to report file name (e.g., to distinguish multiple 'placed' reports)
`[-format <string>]`	Y	Specify report formats; default is { text html csv }
`[-temperature <string>]`	Y	The temperature selection to do timing analysis at a PVT corner
`[-voltage <string>]`	Y	The voltage selection to do timing analysis at a PVT corner
`[-grade <string>]`	Y	The speed grade selection to do timing analysis at a PVT corner

run_tool

`run_tool <id> [-args <string>]`

This command runs a registered tool executable by tool ID, as specified in the ACE extensions config.xml file.

Argument	Optional	Description
`<id>`		The required <id> argument must match a registered tool executable by tool ID, as specified in the ACE extensions config.xml file.
`[-args <string>]`	Y	The optional -args <tool_args> option is used to pass commandline arguments to the underlying tool.

run_un_post_process

`run_un_post_process [-reportsdir <string>] [-debugdir <string>] [-reroute]`

This command removes design post-processing.

Argument	Optional	Description
`[-reportsdir <string>]`	Y	The optional -reportsdir <dir> option is used to override the default location for report files during this step.
`[-debugdir <string>]`	Y	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
`[-reroute]`	Y	The optional -reroute option is used to re-route the affected nets after un_post_process

run_unplace

`run_unplace [-fixed] [-boundary] [-core] [-constants] [-insts <list>]`

Unplace instances in the design

Argument	Optional	Description
`[-fixed]`	Y	Unplace instances with fixed placement constraints as well as movable instances
`[-boundary]`	Y	Only unplace boundary instances
`[-core]`	Y	Only unplace core elements
`[-constants]`	Y	Only unplace constant sources
`[-insts <list>]`	Y	Only unplace the instances specified in this list

run_unroute

```
run_unroute [-net <string>] [-pin <string>] [-nets <list>] [-regions <list>] [-
regionarea <list>] [-nocore] [-clock_only] [-core] [-keepsametile] [-uniqify] [-consts]
[-keepconsts]
```

Remove all or parts of a routing.

Argument	Optional	Description
`[-net <string>]`	Y	Only remove routing for given net
`[-pin <string>]`	Y	Only remove routing for given pin
`[-nets <list>]`	Y	unroute only the nets specified in this list
`[-regions <list>]`	Y	unroute only the nets in the given regions
`[-regionarea <list>]`	Y	unroute only the nets in the given regions
`[-nocore]`	Y	unroute only nets in the I/O-ring
`[-clock_only]`	Y	only unroute clock nets
`[-core]`	Y	only unroute core nets
`[-keepsametile]`	Y	keep all same-tile - connections
`[-uniqify]`	Y	make constants unique again
`[-consts]`	Y	only unroute constants
`[-keepconsts]`	Y	do NOT unroute constants

save_clock_preroute

`save_clock_preroute [-outputfile <string>] [-add_to_project]`

This command will save the pre-routing constraints to a PDC file.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The optional -outputfile <file> option is used to specify the file path to which the add_clock_preroute commands will be saved. If this option is not used, the file will be saved to <project_dir>/<impl_dir>/output/<project_name>_clock_preroutes.pdc
`[-add_to_project]`	Y	The optional -add_to_project option is used to specify that the file should be automatically added to the active project. If this is omitted, the generated PDC file is not automatically added to any project.

save_impl

`save_impl <filename> [-no_log]`

The save_impl command always uses the active impl, since only the active impl is connected to the live DB state. All other impls have no live DB state. The save_impl command saves the state of an impl (impl options and db state) to a . acxdb file. By default, the .acxdb file will save the entire state of the current DB, including placement and routing information. If an impl is saved before running the prepare flow step, a warning message will be printed and only the impl options will be saved, since the DB has not been prepared.

Argument	Optional	Description
`<filename>`		Specifies the ACXDB file path where the active impl state should be saved
`[-no_log]`	Y	If the -no_log option is set, no additional debug information will be saved in the ACXDB file, including log files from the current ACE session

This functionality is also accessible through buttons/menus in the ACE GUI – see Saving Implementations (see page 276) . See also: `restore_impl` and Restoring Implementations (see page 277).

save_partition_placements

`save_partition_placements [-outputfile <string>] [-partition <string>] [-add_to_project]`

Save the partition placement constraints to a PDC file which can be added to the active project.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The name of the output file
`[-partition <string>]`	Y	The name of the specific partition to save.
`[-add_to_project]`	Y	The optional -add_to_project option is used to specify whether the generated PDC file should automatically be added to the active project or not.

save_placement

```
save_placement [-iofile <string>] [-corefile <string>] [-add] [-output_regions] [-
create_iopins] [-io_only] [-iopins_only] [-core_only] [-fixed_only] [-fix] [-
device_port_names] [-instances <list>]
```

Save the current placement to one or more files as a set of pre-placement commands

Argument	Optional	Description
`[-iofile <string>]`	Y	The -iofile <file> option is used to specify the file path to save the IO Ring Boundary instance pre-placement commands to. If this option is not used, the file will be saved to the active project's directory as io_preplacement.pdc.
`[-corefile <string>]`	Y	The -corefile <file> option is used to specify the file path to save the Core instance pre-placement commands to. If this option is not used, the file will be saved to the active project's directory as core_preplacement.pdc
`[-add]`	Y	The -add option specifies that the outputfile should be automatically added to the active project's constraints. (It will be added to the end of the constraints list, and will thus be the last constraints file loaded.)
`[-output_regions]`	Y	The -output_regions option is used to enable output of region constraints into the Core PDC file (or IO Ring Boundary PDC file if -io_only is used)
`[-create_iopins]`	Y	The -create_iopins option is used to enable output of create_boundary_pin commands into the IO Ring Boundary PDC file
`[-io_only]`	Y	The -io_only option is used to specify whether only the IO Ring Boundary pre-placement file is output or not
`[-iopins_only]`	Y	The -iopins_only option allows you to save only the placement of the boundary pin instances
`[-core_only]`	Y	The -core_only option is used to specify whether only the Core pre-placement file is output or not
`[-fixed_only]`	Y	The -fixed_only option is used to specify whether only Fixed Instance placement data is output or not
`[-fix]`	Y	The -fix option forces all saved placements to be "fixed" placement, allowing you to lock down all of your placed instances
`[-device_port_names]`	Y	The -device_port_names option is used to specify whether device port names should be output for IPINs/OPINs instead of site names

Argument	Optional	Description
`[-instances <list>]`	Y	The -instances <instances> option is used to specify a specific list of design instances you want to save placement for. You can call the find command to pass its results to this option.

save_project

`save_project [-project <string>] [-outputfile <string>] [-acxdb <string>] [-no_log]`

The save_project command saves the state of an ACE project to a .acxprj project file. By default, the active project is saved, unless the -project option is specified. The project is saved to the original project file path unless -outputfile is used. If the project contains the current active impl, the state of the DB can optionally be saved to an .acxdb file using the -acxdb option.

Argument	Optional	Description
`[-project <string>]`	Y	The optional -project <projectName> option may be used to specify a project to write out.
`[-outputfile <string>]`	Y	The optional -outputfile <projectFile> option may be used to specify an output file location.
`[-acxdb <string>]`	Y	Enables output of an ACXDB file for the active implementation and requires a file path to save the state of the active impl to
`[-no_log]`	Y	If the -no_log option is set, no additional debug information will be saved in the ACXDB file, including log files from the current ACE session

save_properties

`save_properties [-outputfile <string>] [-add]`

This command is used to save all changed properties on objects in the DB after prepare has been run.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The optional -outputfile <file> option is used to specify the file path to which the set_property commands will be saved. If this option is not used, the file will be saved to the active project's directory as properties.sdc
`[-add]`	Y	The optional -add option is used to specify that the file should be automatically added to the active project. If this is omitted, the file of changed properties is not automatically added to any project.

save_regions

`save_regions [-outputfile <string>] [-region <string>] [-all] [-explicit] [-add]`

Save the placement region constraints to a file, using a TCL command history list approach by default.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The name of the output file
`[-region <string>]`	Y	The name of the region to save. Saves explicit instance list.
`[-all]`	Y	Save all regions (default)
`[-explicit]`	Y	Save explicit lists of instances constrained to each region, as opposed to saving the sequence of TCL commands used to build up the region constraints
`[-add]`	Y	The optional -add option is used to specify whether the file automatically added to the active project or not.

select

```
select <objects> [-clear]
```

This command is used to control the current object selection.

Argument	Optional	Description
`<objects>`		The required <objects> argument specifies a list of objects to append to the current selection. Objects must be prepended with object type prefixes (see "find" command).
`[-clear]`	Y	The optional -clear flag is used to deselect all objects before performing the select action.

The ACE GUI provides a graphical interface for this command through the Selection View (see page 136). See also: Object Type Prefixes (see page 305), Search View (see page 132), `find`, Selecting Objects in the Floorplanner (see page 313), `trace_connections`. (see page 607)

set_active_impl

```
set_active_impl <implName> [-project <string>]
```

This command sets the active implementation for the current ACE session. The active implementation controls which implementation the flow and project management commands are operating on.

Argument	Optional	Description
`<implName>`		The required <implName> argument is used to specify the name of the implementation to set as the active implementation.
`[-project <string>]`	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the active implementation to be set in.

set_clock_type

```
set_clock_type <clock> [-batch] [-boundary] [-trunk] [-minitrunk] [-data_region] [-
data_center] [-data_local]
```

Set properties of a clock. If a non-driving (target) clock pin is specified, a local property is set for that pin only.

Argument	Optional	Description
`<clock>`		net or pin ('inst/pin')
`[-batch]`	Y	Postpone application of this constraint until apply_placement is called at the end of run_prepare. This is useful for nets that are created or renamedby ACE during run_prepare
`[-boundary]`	Y	boundary clock (not routed through the trunk)
`[-trunk]`	Y	trunk clock
`[-minitrunk]`	Y	routed via minitrunk instead of main trunk
`[-data_region]`	Y	data as clock, using region resources
`[-data_center]`	Y	data as clock, using center resources
`[-data_local]`	Y	data as clock, using local resources

set_cluster

```
set_cluster <cluster> [-id <int>] [-wt <int>]
```

Pre-placement command to generate user-defined clusters. This clustering command directs the ACE Placer to keep the specified instances together.

Argument	Optional	Description
`<cluster>`		The required <cluster> list argument is a list of instances which should be clustered in an RLB half
`[-id <int>]`	Y	Optional 2nd level cluster id
`[-wt <int>]`	Y	Optional 2nd level cluster weight {2, ..., 5} - higher weights signify cluster importance

set_equivalent_pins

```
set_equivalent_pins <equivalent_pin_names>
```

This command marks multiple nets or instance pins as functionally equivalent

Argument	Optional	Description
`<equivalent_pin_names>`		A list of pin names or net names that are to be marked as functionally equivalent (<p:toplevel_port_name> | <t:user_pin_name> | <n:net_name>)

set_flyline_direction

`set_flyline_direction <direction>`

This command is used to control whether selected instance flylines are shown for loads of the selected instance, drivers, or both.

Argument	Optional	Description
`<direction>`		The required <direction> argument specifies whether selected instance flylines are shown for loads of the selected instance, drivers, or both. Valid values are loads_only, drivers_only, both and all

set_impl_option

`set_impl_option <option_name> [<value>] [-project <string>] [-impl <string>]`

This command sets options for a project implementation. Only one option may be set at a time.

Argument	Optional	Description
`<option_name>`		The name of the impl option to set a value for. To see a list of valid impl options, use the report_impl_options TCL command.
`[<value>]`	Y	The new value to set the impl option to.
`[-project <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.
`[-impl <string>]`	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.

set_max_flyline_fanout

`set_max_flyline_fanout <limit>`

This command is used to hide selected instance flylines for nets with fanout greater than the limit passed in.

Argument	Optional	Description
`<limit>`		The required <limit> argument specifies the maximum fanout for flylines to be displayed for a net.

set_partition_force_changed

`set_partition_force_changed <name> <changed>`

Set the force changed flag of a partition with the given name. This command should be called at the end of the incremental compile flow. Setting the flag will force the partition to be recompiled the next time that the flow is run, even if the partition timestamp has not changed.

Argument	Optional	Description
`<name>`		Name of the Partition
`<changed>`		Set to 1 to mark this partition as force changed, set to 0 to reset the flag and mark the partition as not forced changed

set_partition_info

```
set_partition_info [-name <string>] [-view <string>] [-cp_type <string>] -timestamp
<string> -import <string> [-comment <string>] [-exclusive_placement]
```

Set partition (compile point) information on the design. This command creates a new partition definition if one does not already exist in the design. This usually comes from the synthesis tool in <design>_partition.tcl

Argument	Optional	Description
`[-name <string>]`	Y	Hierarchical pathname of the partition in the netlist (e.g. /top_module/instance1 /instance2)
`[-view <string>]`	Y	Module name of the partition in the original RTL. Will be used as the name of the blackbox module if the partition is exported.
`[-cp_type <string>]`	Y	Compile point type (hard, locked, or soft)
`-timestamp <string>`		Timestamp in seconds to indicate when this partition was last synthesized. Default is current time in seconds. The options -timestamp and -import are mutually exclusive
`-import <string>`		Pathname to the .epdb file containing the placed and routed partition database to be imported. The options -timestamp and -import are mutually exclusive
`[-comment <string>]`	Y	User specified comment string describing the partition
`[-exclusive_placement]`	Y	Sets the given partition with an exclusive placement property. Instances belonging to the current partition will not be placed together with non partition instances

set_placement

```
set_placement <objName> <siteName> [-fixed] [-batch] [-partition] [-warning] [-
auto_place_neighbors <int>]
```

This command assigns the placement of an instance to a site

`<objName>`		The required <objName> argument is used to specify the name of an instance (i:) or port (p:) to be placed. If multiple objects are to be placed at once, a TCL list of objectnames may be specified. NOTE: Ports (p:) may not be used if the port is connected to a black box IO Ring instance. The port (p:) may only be used to place boundary pins (IPIN, OPIN, CLK_IPIN, CLK_OPIN) that are connected to the port.
`<siteName>`		The required <siteName> argument is used to specify the site (s:) to place the instance on. In the case of IO pin placement, a device port name (d:) may be specified instead of a site name. If multiple objects are to be placed at once, a TCL list of sitenames may be specified.
`[-fixed]`	Y	The -fixed option specifies that the placement of the instance should be fixed to this site and not movable by the placer
`[-batch]`	Y	Postpone application of this constraint until apply_placement is called at the end of run_prepare. This option is useful for instances that are created or renamed by Ace during run_prepare.
`[-partition]`	Y	Use this instance as an anchor to specify the placement of an entire partition relative to the specified site
`[-warning]`	Y	Instead of erroring out, only print a warning message if the instance(s) specified with <objName> do not exist at the time this constraint is applied
`[-auto_place_neighbors <int>]`	Y	This option can be applied to PINs and takes the number of logic levels to process as an argument. It currently only supports placing LUTs driving OPINs for 1 level of logic.

set_project_constraints_pvt

`set_project_constraints_pvt <file> [-corner <string>] [-temperature <string>] [-voltage <string>]`

This command allows the user to set the specific PVT conditions in which an SDC constraint file is applied.

Argument	Optional	Description
`<file>`		The required <file> argument is used to specify the file path to the SDC constraint file.
`[-corner <string>]`	Y	The -corner <corner> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given process corner. Valid values are "fast" and "slow"
`[-temperature <string>]`	Y	The -temperature <temp> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given temperature corner. Valid values are device-specific and must match a value from the junction_temperature impl option list.

Argument	Optional	Description
`[-voltage <string>]`	Y	The -voltage <v> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given core voltage corner. Valid values are device-specific and must match a value from the core_voltage impl option list.

set_property

`set_property <propName> <propValue> <objects> [-warning] [-quiet]`

This command is used to set properties on objects in the DB.

Argument	Optional	Description
`<propName>`		The required <propName> argument specifies property name to set on the objects passed in.
`<propValue>`		The required <propValue> argument specifies property value to set on the objects passed in.
`<objects>`		The required <objects> argument specifies a list of objects to set the property for. Objects must be prepended with object type prefixes (see "find" command).
`[-warning]`	Y	The optional -warning option allows you to downgrade error messages about missing netlist objects to warning messages
`[-quiet]`	Y	The optional -quiet option allows you to disable printing of info messages

set_region_bounds

`set_region_bounds <region> <bounds> [-snap_to_clock_regions] [-snap_to_fabric_clusters] [-snap <string>]`

This command updates a placement region's bounding box of tiles

Argument	Optional	Description
`<region>`		Name of the region
`<bounds>`		List of bounding box coordinates {x1 y1 x2 y2}. x1 and y1 are the upper left corner of the box. x2 and y2 are the lower right corner of the box.
`[-snap_to_clock_regions]`	Y	Snap the bounding box to clock region boundaries (deprecated, use '-snap clock_regions')
`[-snap_to_fabric_clusters]`	Y	Snap the bounding box to fabric cluster boundaries (deprecated, use -snap fabric_clusters')
`[-snap <string>]`	Y	How to snap the region bounding box coordinates. Legal values are: 'none', 'tiles' (the default), 'fabric_clusters', 'clock_regions'

set_region_type

`set_region_type <region> [-type <string>] [-soft] [-include_routing]`

This command updates a placement region's type

Argument	Optional	Description
`<region>`		Name of the region
`[-type <string>]`	Y	What type of placement region this is. Legal values are: 'inclusive' (the default), 'keepout', 'soft'. Instances added to an 'inclusive' region (and attached routing wires when '-include_routing' is set) will be placed within the region bounding box. An 'inclusive' region permits instances to be placed inside the region even if they do not belong to the region. A 'keepout' region prevents any instances (and routing wires when '-include_routing is set) from being placed inside the region. No instances may be added to a 'keepout' region. Instances added to an 'soft' region will be pulled toward the region's center during placement, but instances are permitted to overflow the bounds of the 'soft' region.
`[-soft]`	Y	A 'soft' placement region attempts to pull instance placement to its center, but allows instance placement to overflow it's bounds
`[-include_routing]`	Y	Constrain routing wires, as well as instances, to stay within the region boundary box

set_units

`set_units`

Set the default units for timing constraints.

sleep

`sleep <seconds>`

sleep for number seconds.

Argument	Optional	Description
`<seconds>`		number of seconds to sleep

source_encrypted

`source_encrypted <tclfile> [-nodigest] [-untar]`

source encrypted tcl file.

Argument	Optional	Description
`<tclfile>`		encrypted tcl file to source
`[-nodigest]`	Y	no digest is also OK
`[-untar]`	Y	encrypted file is tar archive, untar first.

trace_connections

`trace_connections <insts> [-drivers_only] [-targets_only] [-include_clocks] [-include_resets]`

This command is used to find instances that are connected to the list of instances passed in. By default, this command traces to find all (drivers and targets) connected instances, except for those connected via clock or reset pins.

Argument	Optional	Description
`<insts>`		The required <insts> argument specifies a list of instance objects to trace connectivity from. Objects must be prepended with object type prefixes (see "find" command).
`[-drivers_only]`	Y	If you want to select only upstream logic that drives the instances in the current selection, use -drivers_only
`[-targets_only]`	Y	If you want to select only downstream logic driven by the currently selected instances, use -targets_only.
`[-include_clocks]`	Y	To include tracing connectivity on clock nets, use -include_clocks.
`[-include_resets]`	Y	To include tracing connectivity on reset nets, use -include_resets.

Similar to the `find` command, this functionality is especially useful when creating lists as input to other Tcl commands, like `select` and `highlight`.

untar

`untar <tarfile> [-member <string>] [-list] [-lz]`

extract content from tar file.

Argument	Optional	Description
`<tarfile>`		tarfile from which to extract
`[-member <string>]`	Y	member to extract, default: none
`[-list]`	Y	list files in archive
`[-lz]`	Y	uncompress first

write_bitstream

`write_bitstream [-outputfile <string>] [-debugdir <string>] [-reportsdir <string>] [-jam] [-flash] [-flash4x] [-max_size <int>] [-hex] [-pcie] [-cpu] [-cpu_width <int>] [-flash_clock_div <int>] [-two_stage] [-nocompress] [-compress] [-chainfile <string>]`

This command generates a programming bitstream for a fully placed and routed design in STAPL format.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation output directory and is named <design_name>.jam.
`[-debugdir <string>]`	Y	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
`[-reportsdir <string>]`	Y	The optional -reportsdir <dir> option is used to override the default location for report files during this step.
`[-jam]`	Y	The optional -jam option may be used to output an additional jam-file
`[-flash]`	Y	The optional -flash option may be used to output an additional serial flash binary file format output
`[-flash4x]`	Y	The optional -flash4x option may be used to output 4 additional 4x serial flash binary file format outputs
`[-max_size <int>]`	Y	The optional -max_size option may be used to set the maximum size of the bitstream for use with -flash and -flash4x options
`[-hex]`	Y	The optional -hex option may be used to output an additional raw hex file format output
`[-pcie]`	Y	The optional -pcie option may be used to output an additional pcie file format output
`[-cpu]`	Y	The optional -cpu option may be used to output an additional CPU Mode file format output
`[-cpu_width <int>]`	Y	This option controls the bit width of the CPU Mode formatted output file. If you are using the CPU interface in x8 mode, set this value to 8. If you are using the CPU interface in x128 mode, set this to 128.
`[-flash_clock_div <int>]`	Y	This option specifies the Serial Flash clock divider value to be used when programming the chip from Serial Flash
`[-two_stage]`	Y	The optional -two_stage option may be used to output bitstream files divided by user mode
`[-nocompress]`	Y	write output as plain-text file
`[-compress]`	Y	compress output-file

Argument	Optional	Description
`[-chainfile <string>]`	Y	The optional -chainfile <file> may be used to override the chainfile set in the active Impl

write_critical_paths_script

`write_critical_paths_script [-outputfile <string>]`

This command writes a tcl script that may be used for viewing critical paths in the synthesis tool.

Argument	Optional	Description
`[-outputfile <string>]`	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation output directory and is named <design_name>_critical_paths.tcl.

write_netlist

`write_netlist [-outputfile <string>] [-debugdir <string>] [-final] [-compress]`

This command generates a verilog netlist for simulation (same as run_generate_netlist).

Argument	Optional	Description
`[-outputfile <string>]`	Y	Output netlist file name.
`[-debugdir <string>]`	Y	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
`[-final]`	Y	Output DRC-free final netlist
`[-compress]`	Y	Compress output file with gzip

write_partition_blackbox

`write_partition_blackbox <partition> [-library <string>] [-outputfile <string>]`

Command to write a verilog blackbox model for a partition

Argument	Optional	Description
`<partition>`		Export a blackbox netlist for the specified partition as a verilog file
`[-library <string>]`	Y	Specifies the name of the blackbox model library (default is "blackbox")

Argument	Optional	Description
`[-outputfile <string>]`	Y	Specifies the output file name for the blackbox netlist (default is <cellname>_bb.v in the implementation output directory)

write_partition_db

`write_partition_db <partition> [-outputfile <string>] [-include_pr_zone]`

Command to export the place-and-route database for a partition

Argument	Optional	Description
`<partition>`		Export the place-and-route database for the specified partition into an .epdb file
`[-outputfile <string>]`	Y	Specifies the output file name for the partition database (default is partitions/<instname>.epdb in the implementation output directory)
`[-include_pr_zone]`	Y	Search for Placement Region that corresponds to partition and use it in splitting routing for nets

write_tcl_history

`write_tcl_history <outputfile>`

Dump the TCL command history for this ACE session to a file

Argument	Optional	Description
`<outputfile>`		The file to save the TCL script to

Chapter - 5: Troubleshooting

This chapter is intended to cover some areas where users frequently report problems, along with solutions. Your Achronix FAE likely can point you to a more recent FAQ.

This chapter will also cover some known ACE issues, with current workarounds where possible.

ACE Exit Error Codes

The following are some of the known exit codes that can be reported by ACE.

Code	Description	Solution
1	Generic error code; catchall for unexpected problem states.	Contact Achronix Technical Support
2	Invalid command line options for ACE or acx	See Running ACE (see page 259) for a list of valid user command-line options. Contact Achronix Technical Support if problems persist.
3	License failure. ACE was unable to obtain a required license.	See the *ACE License & Installation Quickstart Guide (UG002)* for more details about configuring ACE license management. Contact Achronix Technical Support if problems persist.
5	GUI startup failure: incomplete installation: compatible Java release not found. This error typically indicates that the included version of Java used by the ACE GUI is improperly configured.	Contact Achronix Technical Support
13	GUI startup failure. This typically indicates that there's a problem in Linux with the LD_LIBRARY_PATH environment variable, though it can also occur due to crashes when loading the 64-bit WebKitGTK+ HTML browser.	Linux users: Unset the LD_LIBRARY_PATH environment variable and try running ACE again. Also, ensure there is a compatible 64-bit HTML browser (WebKit or WebKit2 for GTK+2 or GTK+3) installed. If problems persist, contact Achronix Technical Support. See Linux: Incompatible Default Web Browser (see page 624) for more details.
100 / 101	The GUI is attempting to workaround a number of known startup issues through automated forced restarts (which display these exit codes). If restarts fail and the GUI did not start successfully, this is an error.	Contact Achronix Technical Support if the GUI did not start.
values ≤ 136	Various license management error codes from RLM.	See the *ACE License & Installation Quickstart Guide (UG002)* for more details on ACE license management. Contact Achronix Technical Support if problems persist.
201	The GUI detected a socket communication error with the acx backend.	Contact Achronix Technical Support

Code	Description	Solution
202	When the GUI was attempting to exit gracefully (due either to user request or a fatal error), critical errors occurred, forcing the GUI to perform a hard kill of itself.	Contact Achronix Technical Support if problems persist.
203	The GUI is unable to start due to underlying framework errors.	Contact Achronix Technical Support. (This is most likely due to an attempt to execute ACE in an unsupported OS.)
404	ACE was attempted to run on an obsolete OS; the GUI is known to be incompatible, so this is disallowed.	Run ACE on a supported operating system.
504	An error occurred while interpreting the Tcl script file passed to ACE.	The Tcl script contains errors or encountered an unhandled error condition. Either fix the bug in the Tcl script, or enhance the error handling in the Tcl script to better handle/report the error condition. Contact Achronix Technical Support if problems persist.
505	No home directory is defined for the current user. By default, ACE places log files and occasional temp files in locations under the user's home directory, so when no home directory is defined, ACE is unable to proceed safely.	Define a home directory for the userid which starts ACE, or change to a userid with a valid home directory before starting ACE. Consult a local system administrator if necessary.
506	ACE is unable to open a socket connection between the GUI and the acx backend. (Specifically, the acx backend is unable to bind a socket port needed for communications with the GUI.)	See the Troubleshooting section below titled: Startup Error — ACE is Unable to Connect on Port NNNN of Localhost (see page 616)
507	The ACE acx backend detected an unexpected GUI socket closure, likely due to a fatal GUI error, when then has caused the acx backend to exit.	Contact Achronix Technical Support

Duplicate Names for Arrays

If the following error message is seen during the prepare stage of ACE, it indicates the occurrence of a duplicate net name in the RTL. The RTL must be modified to clear the error.

```
ERROR: int_cnt[1] is already declared (VNLR-1044)
```

Note

This situation only occurs when one of the duplicates is a single-dimensional array, and the other is a two-dimensional array.

Clock Definitions/Constraints

At least one clock must be defined. Clocks should not be redefined.

Asynchronous Reset of I/O from the Core

If an I/O is not clocked by a boundary clock, use synchronous reset only.

Multi-process Functionality License Requirements

Multi-process functionality requires a license for each background process; therefore users with a single license cannot access this functionality. When encountering this limit, contact your FAE for current workarounds.

> **Note**
>
> Node-locked licenses support multi-process flows without issues. Floating licenses require a new license for each process.

Non-ASCII Characters in Path

Do not use non-ASCII characters in paths. For example, if the username includes German extended characters (e.g. umlauts), Chinese, etc., ACE might function incorrectly. To remedy this, ensure that all paths only contain ASCII characters.

Unable to Load Project: Project is Locked

Example error message for locked project

```
cmd> restore_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1"
Project: "~/output/quickstart/quickstart.acxprj" is locked by another ACE session and cannot be loaded.
This project is locked by user: Docs on host: hostname. You can use restore_project -force to override
the lock. To manually unlock this project, delete the lock file: ~/output/quickstart/quickstart.lock
cmd>
```

ACE locks the Project File (see page 216)s every time it loads a project to prohibit corruption of the project's data. If project locks were not used, and more than one ACE session was allowed to open the same files (or write to the same files) simultaneously, the results would be inconsistent and project data files could become corrupted.

> **Warning!**
>
> Do not use this forced unlock procedure to run multiple ACE sessions on the same project simultaneously, or file corruption might occur!
>
> Project definitions, Implementation definitions, saved implementation states (for both normal flow and incremental compilation), log files, and output directories might get corrupted from having two or more ACE sessions writing to the same files.
>
> Most notably, do not start another ACE session on a project while Multiprocess is already running on that same project; the Multiprocess session must own the project lock (which also locks all implementations) to ensure consistent results and avoid file corruption.
>
> **Achronix does not support simultaneously running multiple ACE sessions on the same project (directory). This is known to cause problems. Do not do this! The project lock files are there to protect users; do not attempt to bypass them.**

In rare cases, ACE can crash, mistakenly leaving a project in the locked state. The easiest way to unlock a mistakenly-still-locked project (which has just failed to load in the GUI) is the following:

1. Double-check that there are no other legitimate users of the project file, including yourself in another desktop!
2. In the Tcl Console View (see page 144), click on the empty **cmd>** line.
3. Click the up arrow (↑) on the keyboard. Each click of the up arrow moves one step backwards through the Tcl command history. Keep moving backwards through the Tcl command history until the Tcl command which attempted to load the locked project is displayed. The failed command should be a call to `load_project` or `restore_project` .

   > **Note**
   >
   > If you regularly load multiple Projects into ACE at the same time, you might have to go back several commands to reach the one that failed.

4. Move to the end of the Tcl command line (press the **End** key on the keyboard), press the space bar, then add the argument `"-force"` to the end of the command. This argument forces ACE to load (or restore) the project despite the presence of a lock, and this session of ACE re-locks the project, taking ownership using a new lock.

   Example workaround

   ```
   cmd> restore_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1" -force
   ```

5. Press the **Enter** key to issue the Tcl command to load or restore the project.

Changing ACE Font Sizes

Fonts in Views

Some views in ACE (particularly diagrams) allow users to specify the exact font/size which will be used. These are typically defined in the ACE preferences, under **Window** → **Preferences** → **General** → **Appearance** → **Colors and Fonts**. There are categorical groupings for each such view allowing its own font and color definitions.

Views that lack their own specific font definition will inherit the font definitions in the **Basic** category, almost always using the **Dialog Font**. A very few views may also use the **Banner Font** and **Header Font** (also found in the **Basic** category) for limited rendering.

The **Dialog Font** used in most ACE views is directly inherited from the underlying GUI application framework stack, (typically called the **Application Font** in Linux/GTK, or **Message Box Font** in Windows).

The various views within ACE often accept font changes immediately, but, in some cases, it might be necessary to restart ACE to see the font changes inherited from the OS propagate completely as the new Dialog Font throughout the application.

> **Caution!**
>
> It is highly recommended that a plain font (not bold, not italics) be chosen as the **Dialog Font**! There is no font naming standard followed across all OS flavors (or even within a single OS), so these plain font varieties are often called "regular", sometimes "book", and sometimes don't have a particular designation.

Linux:

The config location can vary in every Linux desktop / version / distro. In Linux, ACE uses the GTK+ widgets and fonts, so changes should be restricted to those settings.

For example, in the CentOS 6.x Gnome desktop, it can be configured from the main desktop (not ACE) menu under **System** → **Preferences** → **Appearance** → **Fonts** → **Application Font**.

As another example, in the CentOS 7.x KDE Plasma desktop, it can be configured from the main desktop (not ACE) menu under **System Settings** → **Application Appearance** → **GTK+ Appearance** → **GTK+ Fonts**.

Finally, as a third example, in the RHEL8 Gnome3 desktop, it can be configured from the Gnome Tweaks tool (`gnome-tweaks`), under **Tweaks** → **Fonts** → **Interface**.

Windows:

In Windows 10 and 11, users are no longer allowed direct control of the font at a system-wide level through the **Control Panel**. (Though some third-party tools still allow this to be configured). Windows users will be best served by changing the **Basic** → **Dialog Font** directly.

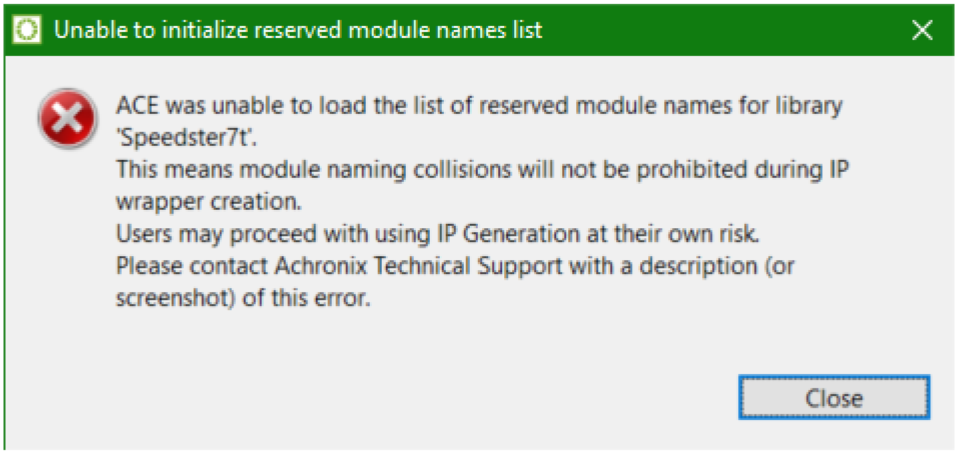
Fonts in HTML Reports

The font used in the HTML Reports is directly inherited from the system's HTML browser's font settings. The system HTML browser can be Internet Explorer (Windows) or WebKit2GTK3 (Linux). The HTML reports typically use the "Proportional" (sometimes called "Web Page") font and Monospace (also called "Plain Text") font types — change these settings in the system's HTML browser to make the fonts change in the ACE HTML Report viewer.

Unable to Initialize Reserved Module Name List

This problem is reported by the ACE GUI at startup with the following dialog.

> **Note**
>
> The exact library name, "Speedster7t" in the screenshot, differs based upon the customer's specific license /installation.



Typically this error indicates that the device overlay files have been installed incorrectly. The list of reserved module names is contained in the overlay files for each Achronix device/library. In very rare cases, the file may exist in the expected location but ACE might lack read permissions for the file.

The file ACE is attempting to read should be located at:

```
<ace_install_directory>/libraries/<library_name>/reserved_module_names.txt
```

where *<ace_install_directory>* is the directory where ACE has been installed and the `ace` executable is found, and *<library_name>* is the name of the library from the error dialog (though forced to all lowercase letters).

Thus, if the Linux user "tester" has installed ACE in their home directory under

`/home/tester/ace/Achronix-linux`

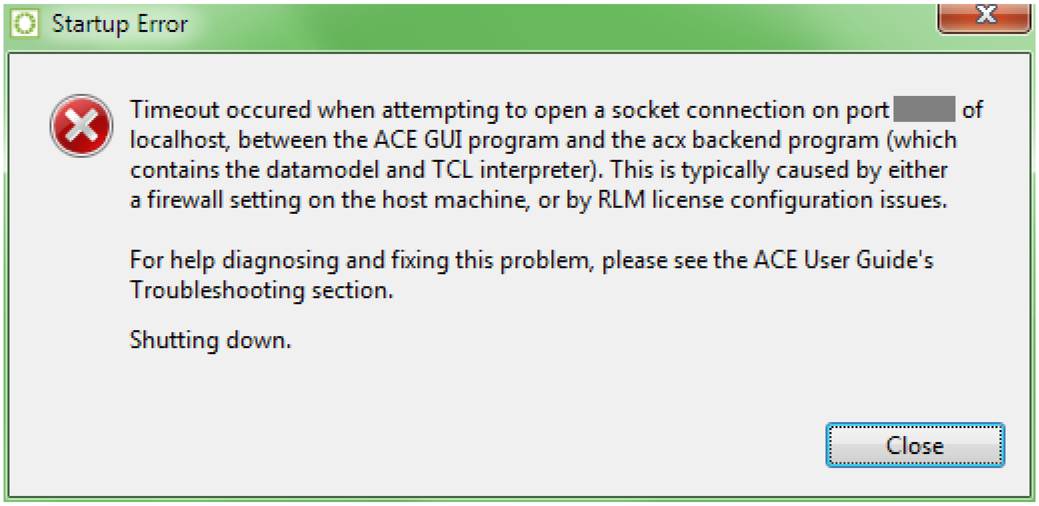
and saw the error dialog captured above about the Speedster7t library, then ACE was unable to find (or potentially experienced permission issues when attempting to read) the file:

`/home/tester/ace/Achronix-linux/libraries/speedster7t/reserved_module_names.txt`

To solve the problem, contact Achronix technical support for help on correctly installing both ACE and the associated device overlay file(s) in the appropriate location.

Startup Error — ACE is Unable to Connect on Port NNNN of Localhost

This problem is reported by the ACE GUI at startup with the following (rather verbose) "Startup Error" dialog:



To Determine the Root Cause

Attempting to run ACE in batch mode (with `ace -batch`, see Running ACE (see page 259)) can help determine the source of the problem. In batch mode, the GUI is not used; therefore, no socket connection is needed between the GUI and the acx backend.

If batch mode ACE does not start, or takes more than 60 seconds before providing the Tcl command prompt (regardless of whether or not ACE reports license errors), then there's typically a licensing configuration problem.

If batch mode ACE starts successfully in less than 60 seconds, then there is a firewall configuration problem.

In very rare cases, usually when ACE is installed in a remote NFS location with many device overlays, poor filesystem (HDD) or network (NFS) performance may also cause these timeouts. In these rare cases, simply trying to start ACE in GUI mode a second time will often succeed where the first attempt had a timeout, because for the second attempt the information will now be found in the filesystem cache in memory, which is much faster to access.

> **Caution!**
>
> Though unlikely, realize it is possible for multiple causes to occur for the same user. It might be necessary to first fix a firewall problem, and then fix a licensing problem. After attempting one kind of fix, if the dialog continues to appear, re-diagnose the problem to verify whether the same issue is still occurring, or whether a new issue is occurring but showing the same symptom.

> When batch mode has started, you can exit ACE batch mode by typing "`exit`" or "`quit`" and pressing Enter.

If it is a Firewall Problem

ACE always uses localhost (IPv4 address 127.0.0.1) TCP sockets to communicate between the ACE GUI program and the ACE backend program `acx` (the pure textual TCL interface when ACE is started with '`ace -b`' or '`ace -batch`'; see Running ACE (see page 259) for details). In most cases, the required localhost sockets are already configured to be available, but in highly restricted network environments, the required localhost sockets might need special permissions to be added to the firewall of the workstation running ACE. Obtain help from your local system administrator and/or network administrator to properly configure the firewall to allow the necessary network traffic.

> **Note**
>
> If the system/network administrator needs to know exactly which executables require firewall permissions, tell them:
>
> (Windows)
>
> - `<ace_install_dir>\system\gui\ACE_GUI_Launcher.exe`
> - `<ace_install_dir>\system\cmd64\acx.exe`
>
> (Linux)
>
> - `<ace_install_dir>/system/gui/ACE_GUI_Launcher`
> - `<ace_install_dir>/system/classic_gui/ACE_GUI_Launcher`
> - `<ace_install_dir>/system/cmd64/acx`

By default, ACE uses an automatically chosen unused (free) TCP port socket somewhere in the range 1024-65535. This automatically chosen port number can change every time ACE is started.

To force ACE to use a specific TCP port number (and thus require only a single hole poked in the firewall for each executable), add the "`-acxport` *<portnumber>*" commandline option when starting ACE (where *<portnumber>* is the desired TCP port number in decimal). Ask your network or system administrator exactly which port number should be specified — this is the same port number the administrator opened for each executable in the firewall.

Please contact Achronix Technical Support if problems persist.

> In some licensing configurations, additional firewall ports might need to be opened specifically for the licensing software. Refer to the *ACE Installation and Licensing Guide* (UG002) for details.
>
> The following executables may require firewall access for the licensing software:
>
> (Windows)
>
> - `<ace_install_dir>\system\cmd64\acx.exe`
> - `<ace_install_dir>\ace.exe`
>
> (Linux)
>
> - `<ace_install_dir>/system/cmd64/acx`
> - `<ace_install_dir>/ace`

If it is a Licensing Problem

A socket connection timeout can also occur when the GUI is attempting to open a connection to the acx backend before the acx backend is ready. This situation can occur if the acx backend is having trouble finding licenses for ACE itself, or for the FPGA/eFPGA devices currently installed for ACE. This long, slow license search is most often seen when ACE is configured to find its license on one or more license servers, and one of those specified servers is not actually a license server (typos and license server migrations being the frequent causes).

Refer to the *ACE Installation and Licensing Guide* (UG002) for details on how to determine the workstation's current RLM license configuration, and how to diagnose and fix what might be going wrong. Frequently, it is simply a matter of correcting or removing the incorrect license server setting in the "`RLM_LICENSE`" environment variable. Contact Achronix Technical Support if problems persist.

Workaround: Extending the Timeout

This workaround is not generally recommended, but in cases where there are transient networking or license server problems, it might be necessary to extend the ACE socket initialization timeout value to be more permissive. To extend the timeout, set the environment variable "`ACX_GUI_INIT_TIMEOUT_SECONDS`" to a decimal number > 60. Invalid numeric values cause a non-fatal warning to be logged, and the default value of 60 seconds will be used instead. Please work with your IT department for help configuring the environment variable such that it will be set for every ACE startup.

If it is a Filesystem or Network Performance Problem

Again, this is rarely the case – licensing configuration problems are much more likely. But if it proves to be a performance problem, installing ACE locally, particularly on an SSD (instead of an HDD), will eliminate network filesystem performance problems. If a local installation is impossible, there are some potential workarounds available.

Workaround: Extending the Timeout

It might be necessary to extend the ACE socket initialization timeout value to be more permissive. To extend the timeout, set the environment variable "`ACX_GUI_INIT_TIMEOUT_SECONDS`" to a decimal number > 60. Invalid numeric values cause a non-fatal warning to be logged, and the default value of 60 seconds will be used instead. Please work with your IT department for help configuring the environment variable such that it will be set for every ACE startup.

Alternate Workaround: Multiple Parallel ACE Installs

Alternately, if there are too many device overlays installed at the same time (the exact value of "too many" will vary based on drive and network performance, but so far this problem has only been reported with more than six device overlays installed on a slow NFS mount), users could install ACE in two (or more) separate locations, with a subset of the device overlays applied in each install location. This would reduce the amount of device (and related license) processing that occurs at ACE startup, to the extent that a timeout will no longer occur.

Multiprocess Summary Report Shows "No Timing Results Found" for Successfully Run Implementations with Existing Timing Reports

In cases of high network file system read/write latency, it is possible that the multiprocess system might not find the required timing information within the allowed period after the external process has completed execution (most likely to occur when using external job submission systems). Sometimes the file writes occurring on the remote machine might be cached for a while, and not immediately written to the NFS drive, so that when the ACE Multiprocess system notices that the spawned process has completed, it does not find the needed timing information on the NFS drive. Therefore, the file read attempt times out, and Multiprocess gives up looking.

For these cases, there is a user preference on the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 201) called **Allowed seconds of NFS write latency**. To fix the problem, increase the value of this preference to allow more time between the completion of the implementation's flow process and when ACE gives up looking for the expected timing information for that completed implementation.

Windows: ACE Incorrectly Reports Read/Write File Permission Problems

In some cases, the ACE GUI might report a file permissions error when attempting to read or write a file on a network drive, when the permissions should actually allow the attempted read or write. As a workaround, please move the affected project to a local, non-network drive location.

Windows: ACE GUI Shown as "Not Responding"

In rare cases, when the Floorplanner Perspective is first changed, the ACE GUI window can become solid grey, and the title bar can change to read something like "**ACE - Achronix CAD Environment - *designName* - *implementationName* - (*deviceName*) - (Not Responding)**". When this problem has been reported, it has always been the case that the Floorplanner is taking too long to repaint.

The Windows operating system requires that applications check-in every five seconds, or the application is deemed non-responsive. Non-responsive applications are given a figurative kick-in-the-pants by Windows, and asked to repaint the screen. When the screen paint itself is taking more than five seconds, as can happen with poor Floorplanner Optimization settings, an application can be forced into an effective infinite-loop of paint requests from the operating system.

If, in Windows, it is ever noticed that the ACE GUI is being called non-responsive by Windows (check the application title bar), ACE has most likely entered this looped painting state. To escape this state, change back to the Project perspective (or any other perspective without the Floorplanner view visible), then navigate to the the Floorplanner View Optimizations Preference Page (see page 195), (**Window** → **Preferences** → **Floorplanner View Optimizations**) and ensure that **Enable Incremental Rendering** and **Render large areas as smaller tiled areas** are both enabled for the current design's complexity level.

> **Note**
>
> Both are enabled by default for everything except trivial designs. Press the **Restore Defaults** button to return to the default settings. If both are already enabled, and the non-responsive state still occurs, please call Achronix Technical Support for guidance on further Floorplanner Optimization tweaks.

Windows: Garbage sometimes appears in the Floorplanner View during panning operations (and remains after panning is completed)

In some cases customers are reporting Floorplanner render errors during/after panning operations.

While Achronix works to fix these rendering errors, there are two potential workarounds to mitigate the render problem.

- Whenever render errors appear in the Floorplanner, first ensure the Floorplanner View has the application focus, then press the backtick (`) key on the keyboard to force the Floorplanner to perform a complete repaint of the full render area.
- In the Floorplanner preferences (**Window** → **Preferences** → **Floorplanner View Optimizations**) enable the **When panning, show only background layer** checkboxes for both the **High** and **Medium** complexity columns. Now, when the panning operation completes, the full view will be automatically re-rendered, eliminating any mid-pan render errors.

Windows: ACE Startup Error Due to Missing DLL Component in Windows 10

In some Windows 10 configurations, the following error dialog might appear when invoking the ACE GUI. This error occurs due to a missing DLL component from the Visual Studio redistributable installer. This situation can be resolved by reinstalling the `vc_redist.x64.exe` executable. This executable can be downloaded from the following link: https://www.microsoft.com/en-ca/download/details.aspx?id=48145.

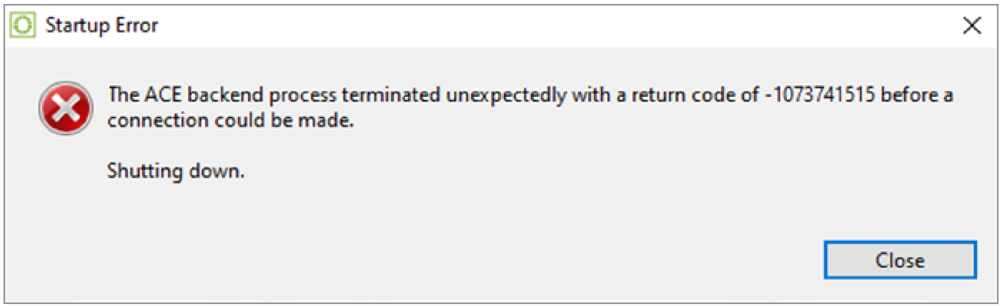


Figure 297: ***ACE Startup Error***

Windows: The icons and buttons in ACE are too small

There are two main ways to deal with this. You can ask Windows to scale everything, which will affect all applications (not just ACE), and if that's not good enough, you can additionally make just ACE further alter the icon (and potentially font) scaling.

Asking Windows to upscale images and fonts for all applications

See the Microsoft documentation here for related information: https://support.microsoft.com/en-us/windows/make-text-and-apps-bigger-c3095a80-6edd-4779-9282-623c4d721d64

Asking just ACE to upscale images and fonts

ACE already scales fonts (and to a lesser extent images) according to what Windows tells it to do, following the settings from the OS, as described above. But by default ACE uses a very coarse scaling granularity for images, and only doubles or triples icon sizes (like 200% or 300%), and does not upscale images/icons by smaller fractions (like 125% or 150%). This is because the fractional scaling can reportedly cause the (Eclipse) frameworks underlying ACE to crash with some video drivers. Thus to maximize stability, ACE disables framework-based fractional scaling by default.

Windows itself has more advanced scaling in the Compatibility Settings area which some users may prefer, but this can also make fonts and images blocky or blurry, so ACE does not enable these settings by default either.

Users having trouble due to the icons/buttons being too small may choose to try either alternate scaling option below (or both in combination) ***at their own risk*** .

Asking Windows to alter the scaling settings for just ACE (which may make text and fonts blurry/blocky)

Close ACE if it is running. Open **Windows Explorer** and navigate to the directory where ACE was installed. (By default this will be `C:\Program Files\Achronix CAD Environment)` In that directory find the `ace` executable, right-click that file, and select **Properties**. A dialog titled **ace.exe Properties** will open. Turn to the **Compatibility** tab, and near the bottom of the dialog press the **Change high DPI settings** button, which will cause a new smaller dialog to appear, also titled **ace.exe Properties**. Near the bottom of this dialog select the checkbox **Override high DPI scaling behavior. Scaling performed by:** and then in the combobox below, select the value **System (Enhanced)** instead of the default value of **Application**. Press the **OK** button to close the small properties dialog, and then press the **OK** button to close the larger properties dialog as well. Start ACE, and observe that image and font scaling may now occur in a more granular /fractional fashion, though things may now be blurry and/or blocky.

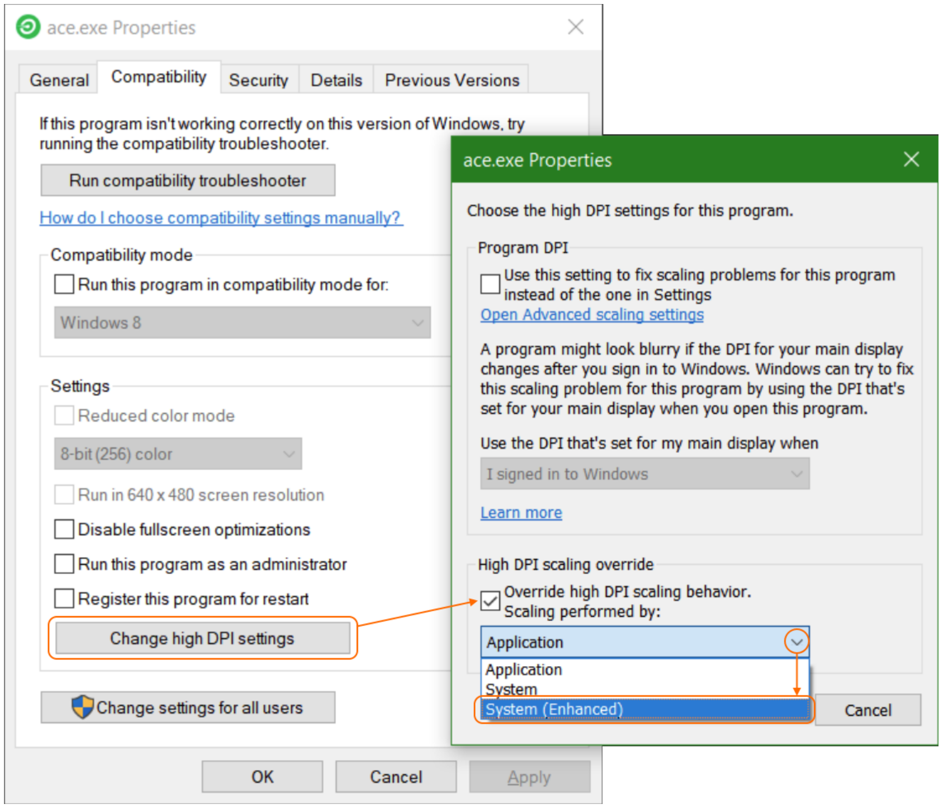


Figure 298: Screenshot showing the sequence to change the DPI settings

To revert this change, disable this supplemental scaling functionality and return ACE to the default behavior:

Close ACE if it is running. Open **Windows Explorer** and navigate to the directory where ACE was installed. (By default this will be `C:\Program Files\Achronix CAD Environment`) In that directory find the `ace` executable, right-click that file, and select **Properties**. A dialog titled **ace.exe Properties** will open. Turn to the **Compatibility** tab, and near the bottom of the dialog press the **Change high DPI settings** button, which will cause a new smaller dialog to appear, also titled **ace.exe Properties**. Near the bottom of this dialog change the setting **System (Enhanced)** back to the default of **Application**, and then deselect (uncheck) the checkbox **Override high DPI scaling behavior. Scaling performed by:**. Press the **OK** button to close the small properties dialog, and then press the **OK** button to close the larger properties dialog as well. The next time ACE is started the scaling will be back to the default behavior.

Enabling ACE's application framework's fractional scaling (fonts remain crisp, but icons may become blurry and ACE may experience instability with some video drivers)

Close all running instances of ACE. Open the file <*ace_installation_directory*>`\system\gui\ACE_GUI_Launcher.ini` in a text editor (like notepad), and at the bottom of the file add a new line:

```
Add the following new line to the bottom of the file ACE_GUI_Launcher.ini

-Dswt.autoscale=quarter
```

And then save the changes to the file. Note that saving the file may require Administrator access; work with IT or MIS personnel to make this change if required.

The next time ACE is started, this will allow ACE to scale images and icons to the more granular fractional values requested by Windows, like 125%, 150%, 175%, 200%, 225%, etc.

If after making this change, some images within ACE vanish, or if ACE experiences crashes (or even silently vanishes), simply remove that new line from the bottom of the `ACE_GUI_Launcher.ini` file to return to the prior (much more stable) behavior.

Linux: Resource Limits: ACE Reports an OutOfMemory Error, But There is Plenty of Free Memory Available

> **Note**
>
> When an OutOfMemory error is reported by ACE, always verify that there is sufficient physical and virtual memory available to run ACE. Running out of physical or virtual memory is the true cause of the error message in >95% of the cases reported.
>
> Consult an Achronix FAE if the ACE memory requirements are unknown for the available licensed Achronix target devices.

There are several types of resource exhaustion in Linux that can be reported as an OutOfMemoryError. Insufficient thread and/or file resources may have similar error reports in some cases.

In some OS configurations, Linux can create a new thread for each file opened, so even when thread resource limits are mentioned in the detailed error message, it could be a case where the file limits (max files open simultaneously) are set too low for the user.

A quick fix attempt would be, before starting ACE, close all other running programs which could have files open. If this works, then the file limits are very likely the problem. But even if this doesn't help on the first attempt, the root problem could still be due to file limits.

In the bash shell, (other shells use different, but often similar, commands,) this is typically managed with the '`ulimit`' command. All current ulimit settings can be queried using '`ulimit -a`', or query just the open file limit with '`ulimit -n`'.

To see if a higher file limit helps, if (for example) the current open files value is 1000, try raising it to 2000, then run ACE. To increase the file limit in this way using the bash shell, use the command '`ulimit -n 2000`'. It might be possible to remove the limit entirely with '`ulimit -n unlimited`'. (Again, users of other shells need to use a different command.)

It is highly recommended that these file limit changes be performed under the supervision of the system administrator. System administrators often apply upper bounds for such ulimit assignments, and individual users cannot exceed those upper bounds without system administrator assistance.

If raising the open files limit does allow ACE to launch correctly, the new raised limit should be applied to the user's '`~/.bashrc`' (or similar) files loaded at shell startup, again with help from the system administrator if necessary. (Alternately, if permissions allow it, create a script used to start ACE. In that script, the file limit could be temporarily raised before starting ACE, and then lowered again when ACE completes execution.)

Linux: In the TWM Window Manager, the First Time the ACE GUI is Started After Installation, the ACE Window is So Small Users Might Not See it

Currently, twm is ignoring the ACE GUI's attempts to set its own initial application window size and location. After ACE is installed (and until the window is moved and resized), the ACE window is in the upper-left corner of the screen, with tiny dimensions (we've seen it as small as 7 pixels wide by 7 pixels tall). This tiny window is often not noticed, especially if there already is a minimized application icon in that region of the screen.

When ACE is running in that tiny window, the ACE window can be moved and enlarged in the same manner as with any other running application window in twm.

ACE does not support the twm standard of choosing the application window's position and dimensions at startup with command-line arguments. Instead, ACE remembers the position and dimensions at application shutdown, and the next time the application is started, ACE returns to that same position and dimension from the last ACE session.

Linux: Odd Behavior When Using X DISPLAY Forwarding if the X Client and X Server Are More than One Major Revision Apart

When running the ACE GUI, the host workstation and display workstation must be at most one OS major revision apart (CentOS6 can talk to CentOS7, but CentOS6 should not talk to CentOS8).

RHEL and CentOS only support X DISPLAY redirection across adjacent major operating system revisions. There are known problems when (for example) applications like the ACE GUI are running on CentOS 8 but having their X DISPLAY redirected to a CentOS 6 workstation, or vice-versa. Users attempting to bridge multiple OS revisions in this way see GUI painting errors and mouse handling errors, especially for drag-and-drop operations. Some users have also reported hung GUIs and application crashes when they attempted to host ACE on CentOS 8 and display on RHEL6.

Because the operating system vendor does not support this behavior, Achronix is unable to support it.

Linux: ACE Menus Do Not Show Icons Next to the Action Names

Most actions within ACE are intended to have an associated graphical icon. This icon is able to be displayed in the drop-down menus within ACE. If no icons are displayed next to actions in menus, this behavior is caused by a GTK+2 configuration that has disabled the icons.

To re-enable icons for all GTK+2 applications (not just ACE), the following command should reset the display. As this issue is the result of GTK+2 functionality, and not ACE functionality, this tweak is not officially supported by Achronix.

```
gconftool-2 --type boolean --set /desktop/gnome/interface/menus_have_icons true
```

Linux: ACE Ignores LD_LIBRARY_PATH

In the majority of cases, the ACE GUI crashes when it encounters custom/obsolete libraries through an assigned LD_LIBRARY_PATH environment variable.

Because of this, by default ACE intentionally ignores preassigned LD_LIBRARY_PATH values when it starts.

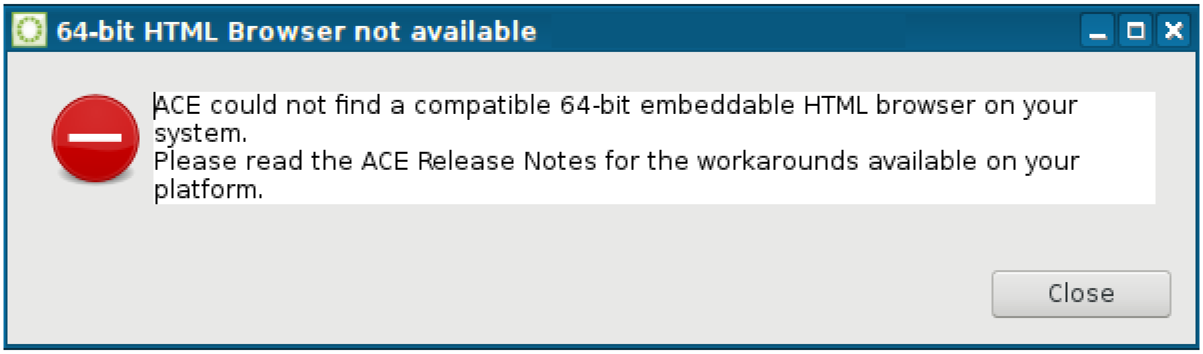
Warning!

Achronix does not support ACE when forced to run using LD_LIBRARY_PATH.

At your own risk, add the command-line argument "`-enable_ld_library_path`" to force ACE to keep the preassigned LD_LIBARY_PATH value at startup.

```
./ace -enable_ld_library_path
```

Linux: Incompatible Default Web Browser



At startup, ACE tries to find a compatible 64-bit embeddable HTML browser already installed on the system for ACE to use to display HTML reports and help content. If no such embeddable browser is detected within the Linux installation, ACE shows the warning dialog above and reverts to a primitive fallback HTML browser (which has slightly reduced functionality and known stability issues on some platforms, but is still better than nothing)

RHEL/CentOS v7.4+ and v8.x customers are not expected to experience any problems with web browser support. Customers running on unsupported Linux distros might need to perform additional steps to ensure ACE has a compatible web browser framework available if the fallback (reduced-functionality) browser proves to be unstable.

Solution

To solve reported web browser incompatibility problems, work with your IT department to install an ACE-compatible 64-bit web browser in your distribution.

For RHEL/CentOS7 (and other GTK+3 Linux distros including RHEL/CentOS8), any WebKit2GTK packages compiled for GTK+3 support are expected to work, regardless of version number, though the latest versions from the official distro repositories are expected to be the fastest and most stable.

Warning!

ACE Linux web browser support requires GTK+3 and WebKit2

Starting in ACE v8.4, ACE is no longer compatible with GTK+2. In Linux ACE requires GTK+3.

Starting in ACE v8.4, ACE is no longer compatible with WebKit (unofficially also known as WebKit1). In Linux ACE requires WebKit2 (the successor to WebKit).

Details

GTK+2

Starting in ACEv8.4, ACE no longer supports GTK+2.

GTK+3

For GTK+3 Linux distros (like RHEL/CentOS7 and later), with the assistance of a system administrator, install a GTK+3 compatible version of WebKit2GTK (*not* WebKitGTK). In RHEL/CentOS, these are expected to be either of the following:

- (RHEL/CentOS7) a package named "webkitgtk4.x86_64", which contains WebKit2 compiled for GTK+3
- (RHEL/CentOS8) a package named "webkit2gtk3", which contains WebKit2 compiled for GTK+3

The needed packages are found within the official RHEL/CentOS distribution repositories and are often installed by default.

When Nothing Else Works

When nothing else works, or when all the available GTK+3 versions of WebKit2 fail to be found or start (due to crashes), one more choice exists. An ancient fallback HTML3 browser is shipped within ACE. This fallback browser appears to be stable on supported Linux distros (while using X11), but it has reduced functionality and performance, renders without antialiasing (the fonts often look ugly), the fonts can be illegible (too small) on HiDPI monitors, it occasionally might not show the entire report on the first try (though scrolling the report or resizing the report usually causes it to fully paint), and it does not directly support the Wayland display server (instead of X11).

ACE automatically tries this fallback browser when it cannot find any compatible embeddable browser within the OS. But if it finds a compatible browser that subsequently crashes, ACE might not get a chance to automatically try the fallback browser.

It is possible to force the use of the fallback browser, which can get around crashes in the various WebKit and/or GTK frameworks. Start ACE with the appropriate argument to force the use of the fallback browser.

```
bash example

export GDK_BACKEND=x11
./ace -force_fallback_html_browser
unset GDK_BACKEND
```

```
csh example

setenv GDK_BACKEND x11
./ace -force_fallback_html_browser
unsetenv GDK_BACKEND
```

Note

This fallback browser is NOT an ideal solution. It is always a much better idea to work with the local IT/MIS support team to get the latest stable version of WebKit2 and GTK+3 installed and working together on the Linux workstation instead.

Additional Information

RHEL/CentOS7

RHEL/CentOS7 includes versions of WebKit2GTK+3 (the package "webkit4.x86_64") within the official release repositories. These are compatible with ACE when running on RHEL/CentOS7. These packages are often installed by default, and must be installed for full functionality to be available in ACE.

RHEL/CentOS8

RHEL/CentOS8 includes versions of WebKit2GTK+3 (the package "webkit2gtk3.x86_64") within the official release repositories. These are compatible with ACE when running on RHEL/CentOS8. These packages are often installed by default, and must be installed for full functionality to be available in ACE.

WebKit and WebKitGTK Technical Details

There are two main APIs for WebKit development. These are known as WebKit2 (the latest edition), and its predecessor known as WebKit (also sometimes referred to for clarity as WebKit(1) or WebKit[1]). There are many released versions of each, including (confusingly) WebKit(1) version 2.*x*.*y*, which is incompatible with WebKit2.

Versions of WebKit and WebKit2 can be compiled to support GTK+2 or GTK+3, which are then all grouped under the WebKitGTK name/prefix in RHEL/CentOS7. (Other Linux distributions including RHEL/CentOS8 have chosen a clearer delineation by using the names WebKitGTK and WebKit2GTK.)

Over time, WebKitGTK has been made available as several package names for RHEL/CentOS, corresponding to the various combinations of support for GTK+2/GTK+3 and WebKit(1)/WebKit2.

- `webkitgtk`: WebKit(1) compiled for GTK+2 (found in non-official community repositories for RHEL/CentOS7)
- `webkitgtk2` : WebKit2 compiled for GTK+2 (only briefly available for RHEL/CentOS7; this appears to have been a short-lived option during the *very* early development of WebKit2)
- `webkitgtk3`: WebKit(1) compiled for GTK+3 (found in the official RHEL/CentOS7 repositories)
- `webkitgtk4`: WebKit2 compiled for GTK+3 (found in the official RHEL/CentOS7 repositories; this is the WebKitGTK package most actively developed/supported for RHEL/CentOS7)
- `webkit2gtk3`: WebKit2 compiled for GTK+3 (found in the official RHEL/CentOS8 repositories; this is the WebKitGTK package most actively developed/supported for RHEL/CentOS8)

Other (Unsupported) Linux Distros

> **Note**
>
> **For unsupported Linux distros:**
> Alternate distributions of Linux are likely to use different package naming schemes than those shown above for RHEL/CentOS.

The naming standards for WebKitGTK packages targeting GTK+2, GTK+3, WebKit, and WebKit2 are non-obvious, and are thus difficult to understand. It is unfortunately very easy to confuse versions of WebKitGTK(1) v2.x with versions of WebKit2GTK, even though they're incompatible.

A package management tool (like 'yum' on RHEL/CentOS) is the best way to research these versions/dependencies, as well as perform the eventual package installation.

A website that can assist with the navigation of the confusing WebKitGTK names and versions is https://pkgs.org /download/webkitgtk. At that site, search for "webkit2gtk" in your chosen Linux distribution, find one that works with GTK+3, and then install it with the local package management tool.

Linux: ACE Requires an Unusually Large Amount of Virtual Memory (Due to WebKit2)

In Linux, the Eclipse framework underlying ACE uses the WebKit2GTK+3 HTML browser framework. This framework uses large amounts of virtual memory (sometimes more than 100GB), apparently as part of the browser's javascript security model. At the present time, there appears to be no way for Achronix to ask/force WebKit2 to use less virtual memory in ACE.

Linux: ACE Draws Slowly Onscreen (or Looks Ugly); Can I Change This Using Themes?

Themes

ACE currently always uses the GTK+3 widgets library in Linux, regardless which desktop environment and/or window manager is being used. The GTK+3 widgets support theme customization. Thus the look and feel of ACE can be modified by changing what is called the "application GTK widgets theme/style/appearance" (the exact name varies based on distro/version/desktop) setting in Linux. By default, to ensure accurate rendering and behavior, ACE forces itself to use the Adwaita theme, even if another theme has been explicitly chosen for the Linux desktop.

> **The default GTK+3 theme "Adwaita" is the only theme supported by ACE.**
>
> The Eclipse GUI framework underlying ACE only guarantees correct behavior and appearance in Linux when the Adwaita GTK+3 theme is in use. Thus ACE is also only officially supported in Linux when running on the Adwaita GTK+3 theme. Because Adwaita is the default GTK+3 theme in all supported Linux versions (RHEL7 /CentOS7/RHEL8), this restriction is not expected to be a problem for most users.

At this time, the default GTK+3/Gnome theme "Adwaita" is the **strongly** recommended choice for best results on GTK+3.

Customers running Linux distributions other than RHEL/CentOS might also have good results with whatever GTK+3 Theme is enabled by default with their distro, but this is only recommended if the preferred Adwaita theme is not available.

When trying alternate themes, Achronix requires that GTK+ v3.22 or later (available as of RHEL/CentOS 7.4) be used with ACE, because those GTK+ versions are reportedly the "final, stable" versions of GTK+3, and are thus the most stable. All releases prior to v3.22 were considered by the GTK+ team to be (unstable) development releases, and should thus be avoided/upgraded if possible.

By starting ACE with the command-line argument `-keep_user_gtk_theme`, ACE stops enforcing the usage of the Adwaita theme, and allows the use of alternate system GTK+3 themes.

Changing the GTK+3 Theme

> **"Dark" themes are not supported in ACE.**
>
> At this time, ACE does not support any dark themes. Customers desiring ACE support for dark themes should request "dark theme support" as an ACE feature enhancement so it can be prioritized appropriately.

The exact setting to change the GTK application widgets theme varies by Linux distribution and version, as well as by desktop manager and version.

Under CentOS7 Gnome, for example:
Gnome Tweak Tool → **Appearance** → **Theme** → **GTK+**

Under CentOS7 KDE, for example:
System Settings → **Application Appearance** → **GTK+ Appearance** → **GTK+ Styles** → **Widget Style**

Under CentOS7 MATE, for example:
System → **Preferences** → **Look and Feel** → **Appearance** → **Theme** → **Customize...** → **Controls**

Consult with your local System Administrator for additional details regarding the configuration of GTK+3 and themes for your local Linux installation.

Animations and Other Effects

While desktop, application/window, and widget animations can improve the feel of applications for some users, other users want to avoid the negative performance impacts.

Animations and special effects are not managed by ACE itself. Often these are controlled within the Desktop Window Manager (exact locations of these settings vary significantly, with some settings available only through the manual editing of Linux configuration files), and some animations may vary even with the user's choice of GTK Theme.

Under CentOS7 Gnome, for example, animations settings can be found at:

Tweak Tool → Appearance → Enable animations

Under CentOS7 KDE, for example, multiple animation settings can be found at:

System Settings → Desktop Effects

Consult with your local System Administrator for additional details regarding the configuration of desktop, application, and widget animations and special effects for your local Linux installation.

Linux: Views and Editors Detach when Dragged Instead of Docking in the Workbench

There is currently a known GTK theme bug (***Linux-only***) in the Eclipse application frameworks underlying ACE that causes view/editor tab docking (including tab re-arrangement) to fail when the Help Window is used. This bug can occur even when the Help Window is minimized; some part of the underlying frameworks is remembering the window size /location despite minimization.

This bug currently appears to be dependent upon which GTK Theme is being used by the Linux window manager. (This theme choice is configured outside ACE.) We have not yet heard any reports of the bug being observed when the system default GTK themes (Adwaita on RHEL/CentOS7 and RHEL8) were in use.

There are three workarounds to allow docking when this bug occurs:

- Close the Help Window while performing the view/editor tab movement operations, and then re-open the Help Window when the movements are completed. (Minimizing the Help Window is not enough. The Help Window must be closed.)
- Shrink and move both the ACE window and the Help Window to a size/location where they do not intersect, then change the view/editor tab locations within the Workbench, then restore the desired Workbench and Help window sizes/locations.
- Work with the local Linux system administrator to change the GTK theme being used, or try to update to a newer /patched version of the chosen GTK theme.
  Some versions of the Clearlooks and Glider themes seem most likely to exhibit the problem.

See the previous section titled Themes (see page 628) for more information on choosing an alternative theme in Linux.

Linux: CDE: Dialogs and Wizards Sometimes Appear Behind the Main ACE Window, Especially After Minimize/Maximize

This problem has only been observed at sites running an X server on RHEL/CentOS and the X client on CDE running within SunOS/Solaris. ***This configuration is not officially supported*** . (Achronix does not support running ACE where either the X server or the X client are on anything except supported operating systems. See the release notes accompanying ACE to determine the exact supported OS versions for a given ACE release.)

CDE has known inter-window focus issues when displaying GTK applications using the default CDE configuration. This problem is not unique to ACE, nor is it something over which ACE has any control whatsoever.

As an example, IBM tools also experience similar problems. A good description of a fix for the issue is in the IBM support forums at: http://www-1.ibm.com/support/docview.wss?uid=swg21124274.

Note

Paraphrased workaround from the IBM support forums:

Basically, the problem is caused by an awkward default setting of CDE that allows modal dialogs to be hidden behind other (parent) windows.

To replace this default setting with a more sane one, the following line needs to be added to `$HOME/.Xdefaults`:

```
Dtwm*secondariesOnTop: True
```

After that, reload the Xdefaults and restart the window manager.

Finally, it might be necessary to also update **CDE Style Manager** → **Windows** where "**Allow Primary windows on top**" should not be enabled (uncheck the checkbox).

If this specific workaround from IBM tech support does not solve the problem in the local CDE configuration, please perform a web search (using similar terminology) with the assistance of a local system administrator to find and apply the fix/workaround for the local Linux window manager configuration.

Warning!

Achronix does not support running ACE on any combinations of Solaris/SunOS/CDE. Consult with a local System Administrator before making these or similar changes on SunOS or Solaris or CDE.

Linux: "Failed to create the part's controls": Some Views and IP Editors may fail to initialize

When this problem occurs, an error message will appear similar to the following screenshot:

The problem has only been observed on older versions of RHEL/CentOS7 (versions prior to RHEL/CentOS7.7), and only with specific Desktop or Window Managers, and only at very high resolutions (when GTK+3 font scaling may become enabled). The error currently appears to be related to bugs in the various graphical library dependencies provided by Linux itself (these are not shipped within ACE).

All customers experiencing the problem have reported that upgrading the version of their preferred Desktop or Window Manager, or changing to another Desktop or Window Manager, or upgrading to newer releases of RHEL/CentOS7 (versions 7.7 or later) has fixed their problem and allowed all ACE Views and IP Editors to once again work as expected.

Upgrading an ACE Installation

Note

This is also supposed to be covered in the *ACE Installation Guide (UG002)*.

On Windows

Achronix presently does not support multiple parallel versions of ACE on the same machine. Thus before upgrading ACE, the prior version should be uninstalled.

1. Disconnect any USB Bitporters
2. (If a node-locked license is being used for ACE:) Copy the `license/*.lic` file from the ACE installation directory to another location (somewhere not under the ACE installation directory).
3. Uninstall the prior version of ACE
4. Install the desired version of ACE
5. (If a node-locked license is being used for ACE:) Copy the `license/*.lic` file back to the proper location within the new ACE installation directory.
6. Re-connect any USB Bitporters
7. Run ACE

Warning!

Installing multiple versions of ACE at the same time is not supported in Windows

Unsupported: Installing multiple versions of ACE at once

This is not officially supported due to limitations in the existing installer/uninstaller framework used by ACE. We do hope to support this scenario in a future ACE release.

Unsupported Workaround:

1. Disconnect any USB Bitporters
2. Install each version of ACE into a separate directory. See the directions below regarding uninstalls.
3. Re-connect any USB Bitporters
4. Run the desired version of ACE.

Be aware that the most recently installed version of ACE is also the first one in the PATH environment variable, which affects the version of ACE and the `acx_stapl_player` that gets executed if/when running those tools manually from the Command Prompt.

Unsupported: Uninstalling ACE after having previously installed multiple versions of ACE at once

This scenario is not officially supported, though we do hope to remedy this in a future version of ACE.

At this time, the ACE uninstaller is only able to uninstall the most-recently-installed version of ACE. (Note that this is not the same as the most recent release of ACE.)

Unsupported Workaround (if the version-to-be-uninstalled is not the version most recently installed):

1. Disconnect any USB Bitporters
2. Re-install the EXACT version of ACE you wish to uninstall on top of itself. ***The installation directory must match exactly.***
3. Uninstall that unwanted version of ACE. When complete, all remnants of that ACE version should have been removed.
4. Repeat steps 2 and 3 (re-install, then uninstall) for each remaining unwanted version of ACE.
5. Re-install the current favorite version of ACE on top of itself. This ensures the favorite version of ACE is once-again the first version in the PATH environment variable (required when running "`ace`" and "`acx_stapl_player`" from the Command Prompt), and also makes the uninstaller once again aware of that version.
6. Re-connect any USB Bitporters

On Linux

Each version of ACE must be installed into a new, empty directory! Never install ACE in the same directory as a prior install.

1. Create a new directory to contain the new version of ACE
2. Untar ACE into the new directory
3. Run ACE

GUI Problems after Upgrading?

In rare cases after an upgrade, (almost always when a different version of ACE is mistakenly installed on top of an existing prior installation,) ACE GUI errors or crashes might occur, especially in the IP Editors.

If you do not wish to perform an uninstall/reinstall of ACE, the following steps often solve the problem:

1. Delete the "`.eclipse`" subdirectory (the leading '.' is important!) in the your home directory.
   - (Windows:) typically "`C:\Users\username\.eclipse\`"
   - (Linux:) typically "`/home/username/.eclipse/`"

   > **Caution!**
   >
   > This subdirectory is hidden in Linux; if unsure what this means or how to find it, please ask your system administrator for help.

2. Try running ACE again

If problems persist:

1. Contact Achronix Technical Support, providing the following log files along with a description of the problem encountered:
   - (Windows):
     a. "`C:\Users\username\.achronix\ace_timestamp.log`"
     where *timestamp* is *year_month_day_hour_minute_second*. (Typically the crash occurred in the most recent log file.) For example:
     "`C:\Users\patsmith\.achronix\ace_2018_12_02_16_06_58.log`"
     b. "`C:\Users\username\.achronix\workspace_version\framework\.metadata\.log`"
     where *version* is the version of ACE which is being run, and *framework* is the Eclipse framework version underlying ACE. For example:
     "`C:\Users\patsmith\.eclipse\workspace_7.1\e46\.metadata\.log`"
   - (Linux)
     a. "`\home\username\.achronix\ace_timestamp.log`"
     where *timestamp* is *year_month_day_hour_minute_second*. (Typically the crash occurred in the most recent log file.) For example:
     "`\home\patsmith\.achronix\ace_2018_12_02_16_06_58.log`"
     b. "`\home\username\.achronix\workspace_version\framework\.metadata\.log`"
     where *version* is the version of ACE which is being run, and *framework* is the Eclipse framework version underlying ACE. For example:
     "`\home\patsmith\.eclipse\workspace_7.1\e46\.metadata\.log`"
2. Delete the ".achronix" subdirectory (the leading '.' is important!) in your home directory.
3. Run ACE

Revision History

<table>
<tr><th>Version</th><th>Date</th><th>Description</th></tr>
<tr><td>1.0</td><td>02 Jul 2016</td><td>Initial Speedcore document release.</td></tr>
<tr><td>1.1</td><td>30 Oct 2016</td><td>
Additions:

Added new tasks detailing Automatic Flop Pushing into I/O Pads (see page 417) and Working with Virtual I/O (see page 425).
Added the page Filter Properties (see page 244) showing all supported filters for the Search Filter Builder Dialog (see page 174), and the find (see page 553), and filter (see page 551) Tcl commands.
Added new Speedster16t IP Configuration Editor sections
Added JTAG support for FTDI FT2232H (in addition to Bitporter)
Added Strict Flow Mode in addition to Evaluation and Normal modes to the Options View (see page 103)
Added new Task content with additional info about Highlighting Objects in the Floorplanner View (see page 315), and updated cross-references for Views providing Highlight functionality.
Added the page Detecting Changes to Project Source Files (see page 297)
Integrated the formerly standalone Incremental Compile Tutorial into this guide, under Using Incremental Compilation (Partitions) (see page 363)

Updates:

Updated the Options View (see page 103) to reflect the latest implementation options.
The Clock Domains View (see page 33), Clock Regions View (see page 36), Partitions View (see page 117), and Placement Regions View (see page 120) have all been updated to mention support for new actions, and dynamic per-device site type columns
Updated the Floorplanner View (see page 53) to reflect the device awareness for Labels and Tooltips in the fly-out palette.
Split the Tcl Command Reference (see page 509) into two sections: SDC Commands (see page 509) and ACE Tcl Commands (see page 534).
Renamed all occurrences of "SnapShot" to "Snapshot"

Removals:

In the Critical Path Diagram View (see page 45), the Layers choices specific to obsolete fabrics have been removed. Associated screenshots and text were be updated.
Obsolete commands were removed from the ACE Tcl Commands. (see page 534)

</td></tr>
<tr><td></td><td></td><td>
Additions:

Concepts (see page 24): Added ECC support to the Speedster16t BRAM Configuration Editor and Speedster16t FIFO Configuration Editor.
Tcl Command Reference (see page 509): Added a new Tcl commands category, Interactive Timing Commands. (see page 528)

</td></tr>
</table>

Version	Date	Description

1.2	30 Nov 2016	**Updates:**
		• Concepts (see page 24):
		  ◦ In the Floorplanner View (see page 53), the choice "IO Port Names" has been re-added to the Labels and Tooltips options for eFPGA/Speedcore products.
		  ◦ Misc flop pushing clarifications in the Options View (see page 103) and Automatic Flop Pushing into I/O Pads (see page 417) pages.
		  ◦ Enhanced the description of strict mode error checking on the Flow Mode (see page 226) page.
		• Tcl Command Reference (see page 509): Clarified when the "p:" port name may be used with the set_placement (see page 603) Tcl command.

1.2.1	23 Dec 2016	**Updates:**
		• Tasks (see page 259):
		  ◦ Minor updates for clarity on Automatic Flop Pushing into I/O Pads (see page 41 417).
		  ◦ Removed references to obsolete *Bitporter and acx_stapl_player Software Release Notes* (RN007).
		• Tcl Command Reference (see page 509):
		  ◦ Added details to get_pins (see page 515).
		  ◦ Corrected error in set_clock_latency (see page 518).
		  ◦ Add details for the option -cpu_width <int> to write_bitstream (see page 60 607).
		• Concepts (see page 24): Removed references to obsolete *Bitporter and acx_stapl_player Software Release Notes* (RN007).
		• Views (see page 31): Added details on the CPU Bus Width option under the Options View. (see page 103)

2.0_beta	01 Feb 2017	**Additions:** • Troubleshooting (see page 611): added section about ACE startup errors regarding localhost TCP ports.

2.0_beta2	28 Feb 2017	**Additions:**
		• Concepts (see page 24):
		  ◦ Added seventeen new Bitstream generation Implementation Options to the Options View (see page 103) for Speedcore eFPGAs (two options specific to Speedster FPGAs were hidden).
		  ◦ Added new preferences to the Project Management Preference Page (see page 210) to disable and/or change the frequency of project source file consistency checks.
		Updates:
		• Tasks (see page 259): Updated Automatic Flop Pushing into I/O Pads (see page 417): the port attribute syn_useiff should no longer be used.
		• Tcl Command Reference (see page 509):

Version	Date	Description
		• add_project_constraints (see page 534) now supports filtering constraints by corner, temperature, and voltage. • run_stapl_action (see page 593) now supports STAPL variable initialization overrides with the new `-defines` option. • write_bitstream (see page 607) now supports configurable bit widths for CPU Mode formatted output files. • Updated help text for get_pins (see page 515), set_clock_latency (see page 518 518), set_equivalent_pins. (see page 601)
2.0	02 May 2017	**Additions:** • Added details regarding Target Device property to the following IP configuration pages: Speedster16t Shift Register Overview Page, Speedster16t ROM Overview Page, Speedster16t DSP FIR Filter Overview Page, Speedster16t BRAM Overview Page, Speedster16t LRAM Overview Page, Speedster16t LRAM FIFO Overview Page, and Speedster16t FIFO Overview Page • Tcl Command Reference (see page 509): • added set_project_constraints_pvt (see page 604)
2.5	01 Jul 2017	**Additions:** • Views (see page 31): Added new Implementation Options "Move Flip-flop Reset", "Pad Flop Pushing Clock Type", and "Report all temperature corners" to the Options View (see page 103). • Tcl Command Reference (see page 509): Added create_boundary_pins (see page 541 541), export_partition (see page 551), get_ace_ext_dir (see page 555), get_ace_ext_li b (see page 555), get_flow_steps (see page 559), get_report_sweep_temperature_c orners (see page 566). **Updates:** • Concepts (see page 24): New IP Configuration Dialog (see page 166) now prohibits IP module name collisions with Achronix's reserved module names. • Concepts (see page 24): The Flow Steps (see page 221) page now includes a table of flow step names and IDs. • Concepts (see page 24): The Speedster16t BRAM Configuration Editor now supports "Simple Dual Port with Soft ECC".
2.9	24 Sep 2017	**Additions:** • Tcl Command Reference (see page 509): Added get_partition_names (see page 562), move_project_netlists (see page 571), write_partition_blackbox (see page 609), wri te_partition_db (see page 610). **Updates**: • Concepts (see page 24): • The Multiprocess Summary Report (see page 238) now supports the Timing Analysis "Report all temperature corners" implementation option, and shows additional columns of timing data for each reported PVT combination. Peak memory usage is now reported alongside the runtimes. Clarity of results is improved in cases where the report contains a mix of Sign-Off timing data for some implementations and Post-Route timing data for other implementations.

<table>
<tr><th>Version</th><th>Date</th><th>Description</th></tr>
<tr><td></td><td></td><td>

The Snapshot Debugger View (see page 139) has been significantly updated to reflect the latest Snapshot Version 3 enhancements.
Configure JTAG Connection Preference Page (see page 186) updated with details regarding multi-device scan chains.

Tasks (see page 259): The entire section regarding Running the Snapshot Debugger (see page 335) has been updated to reflect the latest Snapshot Version 3 enhancements. These include new features like: Startup Trigger; Edge Triggering; configurable monitor, trigger, and stimuli widths; Repetitive Trigger mode; etc. For complete details, see the Snapshot User Guide appropriate to each Achronix device family.

</td></tr>
<tr><td>2.10</td><td>24 Dec 2017</td><td>
Additions:

Concepts (see page 24): Added supplemental information describing Timing Across All Temperature Corners (see page 246)
Tcl Command Reference (see page 509): Added new commands: export_all_partitions (see page 551), generate_route_delay_table (see page 555), insert_delay (see page 570)

Updates:

Concepts (see page 24):

On the Configure JTAG Connection Preference Page (see page 186), corrected misleading information regarding multi-device scan chains, and removed now-obsolete configuration settings (for pod type and connection type).
Removed now-obsolete information from the Options View (see page 103) regarding JTAG configuration settings.
In the Flow View (see page 61), added the Warning icon to the icons table. The icon is primarily used to indicate out-of-sync files, as described in Detecting Changes to Project Source Files (see page 297).

Tasks (see page 259):

The Configuring the JTAG Connection (see page 331) page was updated to further clarify details regarding multi-device scan chains, and to remove mention of now-obsolete configuration settings (for pod type and connection type).
The Generating Timing Reports (see page 325) page was updated with additional information about multiple temperature corners and related Tcl commands.
The Running Multiple Flows in Parallel (see page 282) page was updated to clarify why external job submission systems must use synchronous/blocking commands, and why asynchronous/non-blocking commands can potentially cause serious problems.
The Single-Process Incremental Compile Tutorial (see page 367) was updated for clarity
The Snapshot Design Flow (see page 336) received an updated diagram

Tcl Command Reference (see page 509): Updated create_boundary_pins (see page 541), create_flow_step (see page 541), export_partition (see page 551), run_fpga_download (see page 586), save_properties (see page 599), write_partition_blackbox (see page 609), write_partition_db (see page 610)

</td></tr>
</table>

<table>
<tr><th>Version</th><th>Date</th><th>Description</th></tr>
<tr><td></td><td></td><td>
Troubleshooting (see page 611): enhanced information regarding ACE error codes
</td></tr>
<tr><td>3.0</td><td>19 Aug 2018</td><td>
Additions:

Concepts (see page 24):

Added new page describing various supported ACE Verilog Attributes (see page 240) that can be applied to instances, nets, pin, ports, or other objects in the ACE datamodel.
Described the new Properties View (see page 128)
Added a description of the Implementation Options Report (see page 240)

Tasks (see page 259)

Added content regarding Applying and Checking Properties (see page 329)

Tcl Command Reference (see page 509):

Added new Tcl commands get_clock_regions (see page 557), get_clock_region_bounds (see page 557), get_file_line (see page 559), get_regions (see page 566), get_region_bounds (see page 565), report_coverage (see page 577 577)
Added new Interactive Timing Commands (see page 528) pages for~~check_timing~~and~~report_clock~~. Reminder: these are stand-alone timer commands, enabled only after prepare_sta (see page 530) is run, and disabled after reset_sta (see page 533) is run.

Updates:

Concepts (see page 24):

The~~JTAG Browser View~~was updated to include a new "Word Step" field.
Corrected the Critical Path Diagram View (see page 45) to reflect the removal of the now-obsolete Layers section of the palette
The configuration of the Floorplanner View (see page 53)'s Route Rendering Mode has been moved from the view's fly-out palette to the Floorplanner View Colors and Layers Preference Page (see page 190)
Clarified how the icon in the Flow View (see page 61) relates to Detecting Changes to Project Source Files (see page 297)
Documented the new ease-of-use feature/button ( ) on the Multiprocess View (see page 83) allowing users to re-open a pre-existing Multiprocess Summary Report (see page 238).
For the Options View (see page 103), updated the listings of common options, and improved the content describing the Tcl interactions for viewing /changing implementation options in general
Improved the description of the Power Dissipation Report (see page 228), including clarification of the ramifications of Timing Across All Temperature Corners (see page 246)
The Projects View (see page 125) gained a new action: Reload Project ( ); added a screenshot and description showing how disabled constraints will be greyed out in this view; clarified details regarding the load order of constraint files.

Tasks (see page 259):

</td></tr>
</table>

<table>
<tr><th>Version</th><th>Date</th><th>Description</th></tr>
<tr><td></td><td></td><td>

Clarified details regarding Adding Source Files (see page 272) to ACE projects, how the source file load order may be changed, and the ways in which constraint files may be enabled/disabled (instead of added/removed) for implementations
Mentioned that it is also possible to disable constraint files in implementations, instead of completely Removing Source Files (see page 275) from projects
Enhanced the instructions for Assigning Placement Region Constraints (see page 358)
Renamed the section 'Getting Floorplanner Object Tooltips' to Choosing Floorplanner Object Tooltips (see page 315) for clarity
Enhanced the descriptions of Loading Projects (see page 270) to cover both GUI and Tcl interactions, with new explanations of project locking and lock files.
Added more thorough discussion of the license ramifications of Running Multiple Flows in Parallel (see page 282), and added a cautionary note describing how multiple implementations can unexpectedly generate identical results after upgrading ACE, along with the recommended fix.

Tcl Command Reference (see page 509):

Updated~~report_timing~~with newly supported command line options.
Renamed 'display_rtl' to display_netlist (see page 546) which will better reflect the command's actual functionality

Troubleshooting (see page 611): updated discussion of project locking and when forcing locked projects to load is acceptable; updated descriptions of font management; updated content for ACE startup error diagnosis (firewall vs license issues); added new content regarding missing icons for actions in menus.

</td></tr>
<tr><td></td><td></td><td>
ACE v7.0 is the first combined release, supporting both Speedster FPGA devices and Speedcore eFPGA devices. Rather than parallel releases, there is now a single release, thus the change in numbering schemes (to be more in sync with the higher-versioned Speedster software, which was in the 6.x release sequence).
Additions:

Tasks (see page 259): added several pages describing the ACE help system: Accessing Help (see page 432); added a new page about Detaching Views and Editors (see page 264); added a new page describing Multiprocess Batch Mode (see page 292 292) (using the new run_multiprocess (see page 587) Tcl command), including the new seed sweep functionality
ACE Tcl Commands (see page 534): added new command run_multiprocess (see page 587)
Troubleshooting (see page 611):added a section regarding changing Linux GTK theme and animation settings to affect the render performance as well as the look and feel of the ACE GUI; added a section describing the known (GTK theme-dependent) bug where views/editors may detach (instead of move) while the Help Window is open;added a section about the impacts of Linux resource limits;added a section about the twm Window Manager in Linux;added a section regarding Linux LD_LIBRARY_PATH concerns;added a section about dialogs in the CDE Window Manager

Updates:
</td></tr>
</table>

Version	Date	Description
7.0	07 Dec 2018	• Concepts (see page 24):The Floorplanner View (see page 53) has changed the presentation of routing errors (open connections, open pins, and route overflows); the Critical Path Diagram View (see page 45) now has right-click context menu items similar to the other views within the Floorplanner Perspective; the Options View (see page 103) section was updated to match the latest lists of options, but now includes only those options which are common to all target devices – implementation options which are unique to specific libraries or devices will now be documented elsewhere, including within the on-demand Implementation Options Report (see page 240). • Tasks (see page 259):The Moving and Docking Views and Editors (see page 262) page and Rearranging Tabbed Views and Editors (see page 264) page have been re-titled and have had their content updated to reflect that Editors can now be moved around just like Views. The user feedback has changed during movement and docking, and the descriptions/tables have been updated accordingly. • ACE Tcl Commands (see page 534):added "-verbose" option to: add_region_find_insts (see page 536), add_region_insts (see page 536), create_region (see page 543), and remove_region_insts (see page 576); find (see page 553) added "-warning" and "-error" options; run_generate_fullchip_sim (see page 586) added "-modelsdir" option; set_false_path (see page 522) has improved description of various options; add_region_find_insts (see page 536), add_region_insts (see page 536), create_region (see page 543), and remove_region_insts (see page 576) added a new "-clocks_only" option; remove_region_insts (see page 576) added a new "-flops_only" option; run_insert_holdbuffers (see page 587) added a new option "-io_buffers". • Troubleshooting: (see page 611)the Linux web browser section has been updated to reflect the change from the Mozilla XulRunner to the WebKitGTK+ HTML browser framework **Removals**: • Deleted Concepts and Tasks pages made obsolete by the updated GUI frameworks: "Fast Views", "Opening Perspectives"
7.1	27 Mar 2019	Additions: • Advanced Concepts (see page 240): added a new page describing ECO Commands (see page 247) Updates: • Reports (see page 227): updated information on the Implementation Options Report (see page 240) page to improve clarity and accuracy • Views (see page 31): added info to the Properties View (see page 128) page covering double-click shortcut gestures and context-menu actions; updated the table of Actions to match the latest functionality on the Projects View (see page 125) page • Tasks (see page 259): updated content on the Multiprocess Batch Mode (see page 292) page for improved clarity • Troubleshooting (see page 611): removed obsolete content and updated content for new potential problems and workarounds, now that ACE has added official support for GTK3 and WebKit2 (both new as of this release, see new content for technical details)
		Updates:

Version	Date	Description
7.2	06 Jun 2019	• Tcl Command Reference (see page 509): Updated the descriptions for all SDC Commands (see page 509), and provided/updated usage examples where applicable. To avoid command naming collisions with other tools, updated the names and descriptions for the Interactive Timing Commands (see page 528). • Views (see page 31): updated the Netlist Browser view (see page 89) actions table, adding missing actions and updating icons that have changed; updated theFlo orplanner View (see page 53)to mention the new **Allow Tooltips**toggle checkbox; updated the Critical Paths View (see page 48) to mention the new **Show Clock Paths** action; updated the Multiprocess view (see page 83) page, updating the screen shot and the text to include the new Seed Sweep functionality; updated the Fl ow View (see page 61) page, updating the screen shot and icon images to match recent updates. • Advanced Concepts (see page 240): updated the ECO Commands (see page 247) section, moving each related dialog into its own page and updating screen shots • Troubleshooting (see page 611): updated the section about Themes to explain that ACE will now enforce the usage of the Adwaita theme when running on GTK3.14+ (to maximize performance and stability).
8.0	17 Sep 2019	Additions: • Concepts (see page 24): added pages for the new I/O Designer Toolkit Views (see page 66) and Generate I/O Ring Design Files Dialog (see page 176). • ACE Tcl Commands (see page 534): added new command: `generate_soc_design_files` *(renamed generate_ip_design_files (see page 555)in the 8.1 release).* Updates: • Concepts: The Projects View (see page 125) page removed the obsolete **Remove All Projects** action and added the new actions to **Clone IP** and **Rename IP**. • Tasks (see page 259): The Single-Process Incremental Compile Tutorial (see page 367) content has had its formatting tweaked. • ACE Tcl Commands (see page 534): The interactive timing command report_checks (see page 531) removed the unsupported argument "-input_pins". The interactive timing command reset_sta (see page 533) had a typo in the examples.
8.1	24 Jan 2020	Additions: • Concepts (see page 24): What was previously a single I/O Designer View (with multiple inner tabs) has now been split into multiple independent-but-related views, grouped under the I/O Designer Toolkit Views (see page 66) page. The new pages are for the I/O Core Pin Assignment View (see page 70), I/O Layout Diagram View (see page 72), I/O Package Diagram View (see page 68), I/O Pin Assignment View (see page 69), and I/O Utilization View (see page 67). (As a reminder, these I/O Designer Toolkit views are only relevant for Achronix Speedster7t FPGAs such as the AC7t1500.) • Troubleshooting (see page 611): Added a section dealing with the most common symptom of improperly installed device overlays: Unable to initialize reserved module names list. Updates: • Concepts (see page 24): updated I/O Designer Toolkit Views (see page 66) page with info about cloning and double-clicking.

Version	Date	Description
		• ACE Tcl Commands (see page 534): The command `generate_soc_design_files` was renamed generate_ip_design_files (see page 555). The command run_unroute (see page 596) has gained arguments to deal with regions.
8.1.1	21 Feb 2020	Updates: • ACE Tcl Commands (see page 534): The command run_insert_holdbuffers (see page 587) was updated to support a new optional argument: "-typebased_buffers".
8.2	17 Jul 2020	Additions: • Concepts (see page 24): Added info about 'Add [copies of] IP to another project...' to Projects View (see page 125). Added info about "active editor highlighting" to I/O Layout Diagram View (see page 72). Added info about 'Remap Port/Signal Name' to I/O Pin Assignment View (see page 69), I/O Core Pin Assignment View (see page 70). • ACE Tcl Commands (see page 534): added new commands: get_fanout (see page 51 514), run_multiprocess_iterator (see page 589) Updates: • ACE Tcl Commands (see page 534): the run_multiprocess (see page 587) command has a new flag `-create_option_sets`; the save_placement (see page 597) command can now save placement of just a subset of the design, using the new option `-instances`; the run_insert_holdbuffers (see page 587) command has a new option `-typebased_buffers`. • Concepts (see page 24): The Multiprocess Summary Report (see page 238) is now automatically sorted by quality of results, instead of alphabetically. The Multiprocess View (see page 83) (and run_multiprocess (see page 587)) can now optionally generate customized option sets for any chosen template implementation. The Netlist Browser View (see page 89) now has convenience filters to hide instances of the cell types of boundary pins, power, and ground. The Save Placement Dialog (see page 170) can now optionally accept Tcl lists (or Tcl statement that generate lists) of Instance names whose placements shall be saved, and can be pre-populated from the Search View (see page 132) and the Selection View (see page 136). • Tasks (see page 259): The Single-Process Incremental Compile Tutorial (see page 36 367) has been updated to improve phrasing/clarity.
8.3	16 Dec 2020	Additions: • Concepts (see page 24): created a page for the new Create a SecureShare Zip File Dialog (see page 177) • Tasks (see page 259): created a page for Using the ACE SecureShare Tool to Create a Support Zip File (see page 435) • ACE Tcl Commands (see page 534): added get_best_multiprocess_impl (see page 55 557), report_performance (see page 579), Updates: • Concepts (see page 24): The old "global" option sets have been removed in favor of the improved custom-generated option sets based on design analysis; the Multiprocess View (see page 83) and Active Project and Implementation (see page 221) pages have been updated accordingly. Added more details about Log Files (see page 218) generated during Multiprocess.

Version	Date	Description
		• Tasks (see page 259): Misc clarifications for Multiprocess Batch Mode (see page 292), option set updates for the Attempting Likely Optimizations Using Option Sets (see page 352). • ACE Tcl Commands (see page 534): remove_impl (see page 574) now supports working on a list of impls instead of just a single impl; clarified help text for run_multiprocess (see page 587) and run_multiprocess_iterator (see page 589);
8.3.2	11 Mar 2021	Additions: • Concepts (see page 24): added a page to describe the new NoC Performance View (see page 95) • ACE Tcl Commands (see page 534): added new commands: get_compatible_placements (see page 557),`move_partition` ,~~`process_move_partition`~~, redirect (see page 572) Updates: • Concepts (see page 24): Under ACE Verilog Attributes (see page 240), renamed "ace_useioff" to "syn_useioff"; on the Perspectives (see page 24) page described the new NoC Performance Perspective; • Tasks (see page 259): The Automatic Flop Pushing into I/O Pads (see page 417) page was updated to use the latest Verilog attributes. • ACE Tcl Commands (see page 534): renamed "report_regions" to report_clock_regions (see page 577), the add_project_ip (see page 535) and remove_project_ip (see page 575) commands now require Tcl lists of acxip filenames to improve behavior with paths containing spaces; apply_placement (see page 537) and set_placement (see page 603) now accept partition names (to be used in combination with~~`move_partition`~~); the report_checks (see page 531) command now supports a new option "-unconstrained"; report_impl_options (see page 578) now supports new arguments "-hide_values", "-show_standard", and "-diff_options";

8.5	03 Aug 2021	Additions:
		• ACE Tcl Commands (see page 534): Added new commands create_equivalent_regions (see page 541), remove_project_constraints_pvt (see page 575), and untar (see page 607).
		• Concepts (see page 24): Added a new Advanced Concept description for Fabric Clusters (see page 258).
		Updates:
		• SmartSupport™ has been renamed SecureShare™, affecting several pages.
		• The Create Placement Region Dialog (see page 159) concept and Creating a New Placement Region (see page 355) task have been updated to reflect new functionality, allowing (not just instance placement, but also) routing to be restricted to the placement region.
		• Concepts (see page 24): The NoC Performance View (see page 95) now supports drag-scrolling.
		• ACE Tcl Commands (see page 534):
		    • The get_property (see page 565) command has been updated with a new "-object_type" argument.
		    • The message (see page 571) command now allows console logging support to be toggled on and off.

<table>
<tr><th>Version</th><th>Date</th><th>Description</th></tr>
<tr><td></td><td></td><td>The report_checks (see page 531) command now warns that the report generated by this command may include less information than the regular Timing Report.
The restore_project (see page 584) description has been updated for clarity.
The set_clock_type (see page 600) command now supports a new "-batch" argument.
The set_partition_info (see page 603) command arguments and descriptions have been updated for clarity.
The write_bitstream (see page 607) command has new arguments "-max_size" and "-two_stage".
Troubleshooting (see page 611): Commentary regarding now-obsolete/unsupported Windows7 and CentOS6 (along with the associated GTK2 details) have been removed.</td></tr>
<tr><td>8.6</td><td>20 Oct 2021</td><td>Additions:
ACE Tcl Commands (see page 534): added new commands <code>optimize_tile</code> , <code>regenerate_all_ip_design_files</code>
Updates:
Lots of pages throughout the document have been updated with very minor edits to improve grammar and clarity of phrasing, as well as fixing typos.
Concepts (see page 24): The Views (see page 31) page has been updated to note that the view context menu icon used by all Views has changed from a tiny down-arrow to a vertical elipsis – (most View screenshots have not yet been updated to reflect this icon change).
ACE Tcl Commands (see page 534): The <code>set_partition_info</code> command gained a new argument <code>-exclusive_placement</code>. The <code>set_placement</code> command gained a new argument <code>-auto_place_neighbors</code>. The <code>run_secureshare</code> command removed the argument <code>-no_archive</code> and gained the argument <code>-wizard</code>.
Deletions:
Removed remaining content regarding the (no longer supported) 22i and 16t IP Configuration Editors.
Removed remaining content regarding the (no longer supported) JTAG Browser View and JTAG Diagram View.</td></tr>
<tr><td></td><td></td><td>Additions:
A new Clusters View (see page 41) has been added to the Floorplanner Perspective.
Updates:
Concepts (see page 24): The Properties View (see page 128) has gained an action /dialog to Save Changed Properties. The Create Placement Region Dialog (see page 159) has been updated with new groupings for the available Region Alignments and Region Types, with new options for each. The Floorplanner View (see page 53) can now show Cluster information in the tooltips. The NoC Performance View (see page 95) now includes tooltip information reporting the comparative "blocked" and "transferred" percentages for the "trying" times, and includes a brief table describing the Preferences for this view.</td></tr>
</table>

Version	Date	Description

8.7	25 May 2022	• Tasks (see page 259): The Creating a New Placement Region (see page 355) page has been updated to show the latest Region Alignment (-snap) and Region Type (-type) choices for the associated dialog.
		• ACE Tcl Commands (see page 534):
		    • The create_region (see page 543) command has been updated with new options `-snap [none|tiles|fabric_clusters|clock_regions]` and `-type [inclusive|keepout|soft]`. The former `-snap_to*` and `-soft` options are now deprecated.
		    • The highlight (see page 569) command has added a new option `-clear`.
		    • The set_region_bounds (see page 605) command has been updated with new options `-snap [none|tiles|fabric_clusters|clock_regions]` and `-type [inclusive|keepout|soft]`.

8.8	18 Jul 2022	Additions: • Concepts: Added a page for the new Load Acxdb Dialog (see page 182). Added a page for the new NoC Time Slice View (see page 100). Added a page for the new Pl ot Serdes Diagram Dialog (see page 183). Added a page for the Configure Clock Pre-Routes Dialog (see page 153). • Tasks: Added a new page for Cleaning Projects (see page 279) under Working with Projects and Implementations (see page 268). Added a new page for Plotting Serdes Rx Diagrams using JTAG (see page 439) • ACE Tcl Commands: added new commands add_clock_preroute (see page 534), re move_clock_preroute (see page 573), save_clock_preroute (see page 596), clean_pr oject (see page 538), get_compatible_ordering_codes (see page 557), get_pvt_corn ers (see page 565), is_labmode (see page 570), set_region_type (see page 605) • Note: Early access support for Partial Reconfiguration has been added, relevant documentation will be included in a future release. Updates: • Concepts: The Netlist Browser View (see page 89) has three new boolean toggle filters for the Instance Names column for Constants, Duplicates/Clones, and Feedthroughs. The Clock Regions View (see page 36), Clusters View (see page 41 41), Partitions View (see page 117), and Placement Regions View (see page 120) now display and allow configuration of Clock Pre-Route information when relevant. The NoC Performance View (see page 95) page was updated with the latest changes to the related preferences. The Floorplanner View Colors and Layers Preference Page (see page 190) was updated to reflect the Always Show Highlighted Instances option, as well as the latest locations/names of several preferences. • Tasks: Updated the Running ACE (see page 259) page to reflect the changes to Lab Mode functionality (lab mode now allows a restricted subset of Tcl functionality, specifically the commands necessary for JTAG operations). Added a section describing ACE Startup Arguments to the Running ACE (see page 259) page. Updated Running the Entire Flow (see page 280) to mention the new Load Acxdb Dialog (see page 182) which can appear when a run is canceled. • Troubleshooting: Added a new section describing a "Failed to create the part's controls" error message, along with a workaround. Added a new section about font and image scaling (for high resolution/DPI monitors) in Windows.

Version	Date	Description
		• ACE Tcl Commands: The create_region (see page 543) command has a new -pr_zone argument to designate partial reconfiguration zones. The get_regions (see page 566) commands has a new -verbose argument to print region information. The load_flowscripts (see page 570) command has a new -bitstream argument to load only those scripts relevant to bitstream manipulations. The ~~move_partition~~ command's arguments have changed. Deletions: • ACE Tcl Commands: The obsolete process_move_partition command has been removed; use set_placement -partition instead. Minor enhancements to document style added throughout.
9.0	10 Feb 2023	Additions: • Tasks: Added new section for the Partial Reconfiguration Tutorial (see page 442)(s) • ACE Tcl Commands: Added new commands get_synprj_from_project (see page 567), move_relative_paths (see page 572), run_generate_final_reports (see page 586), save_partition_placements (see page 597) Updates: • Concepts: On the Flow Steps (see page 221) page, added the new flow step named 'Generate Final Reports'. Updated Design Completion Steps (see page 224) page to add description for the new flow step 'Generate Final Reports'. Updated Clock Domains View (see page 33), Clock Regions View (see page 36), Clusters View (see page 41), Partitions View (see page 117), and Placement Regions View (see page 120) with refreshed screenshots and descriptions for the new functionality to support Partial Reconfiguration. Updated the Save Placement Dialog (see page 170) page to reflect the latest options. • Tasks: Updated the Multiprocess task page Running Multiple Flows in Parallel (see page 282) to mention node-locked license management; the page previously only mentioned floating license concerns. • ACE Tcl Commands: add_clock_preroute (see page 534) has a new argument -data_region; add_project_netlist (see page 535) and remove_project_netlist (see page 576) have a new argument -impl; create_boundary_pins (see page 541) has a new argument -purpose; find (see page 553) has improved help text describing pin pattern matching; get_impl_option (see page 559) has a new argument -syn; the help content for report_checks (see page 531) has been tweaked to improve clarity; report_power (see page 581) now has added support for saif files; report_routing (see page 582) has a new argument -wl; source_encrypted (see page 606) has a new argument -untar; write_bitstream (see page 607) has a new argument -pcie; write_partition_db (see page 610) has a new argument -include_pr_zone; • Throughout the document, hyperlinks that point outside the document have been updated to static text to prevent outside content from appearing in the GUI help. Throughout the document, content has been edited to improve clarity and consistency. Deletions: • ACE Tcl Commands: The obsolete move_partition has been removed; use set_placement -partition instead.
		Additions:

Version	Date	Description
9.1	26 Apr 2023	• Concepts: Created a new page describing the Project concept of Port Mapping Files (see page 218) (as used in IO Designer). Created info covering the new I/O Designer Preference Page (see page 199), new Netlist Browser Preference Page (see page 203), and new NoC Performance View Preference Page (see page 203). • ACE Tcl Commands (see page 534): Added new command remap_partial_bitstream (see page 573) Updates: • Concepts: Updated the Save Placement Dialog (see page 170) page to reflect the latest options. Updated the Configure Clock Pre-Routes Dialog (see page 153) to reflect the latest functionality. • Troubleshooting (see page 611): Removed info related to obsolete versions of Windows. Updated Linux usage directions for the fallback html browser (to improve functionality under Wayland). Updated Windows info related to upgrading ACE and multiple parallel installs of ACE. • ACE Tcl Commands (see page 534): Updated the arguments for run_timing_analysis (see page 594); Clarified the usage description of set_partition_force_changed (see page 602). • Throughout the document, content has been edited to improve clarity and consistency.