ACE User Guide (UG001)

Achronix CAD Environment (v7.2)

Achronix

Data Acceleration

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 the Achronix CAD Environment (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 21) includes an Introduction to ACE and a quick Tutorial.
- Concepts (see page 23) covers all the basic concepts of ACE, and can be considered a reference manual for the various GUI elements.
- Tasks (see page 344) details how to complete various tasks within the GUI, plus provides the related TCL commands.
- TCL Command Reference (see page 510) provides a complete TCL command reference, including syntax.
- Troubleshooting (see page 602) shows a number of common problems and the recommended solutions.
- Revision History for ACE for Speedster FPGAs (Working) lists the changes to each revision of this document.

Related Documents

The following documents, as well as the latest version of this document (UG001), will always be 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)
- *ACE STA Constraint User Guide* (UG053)

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 will 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 238), click the **Create Project** ( ) toolbar button. In the Create Project Dialog (see page 273), 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 351) or Working with Projects and Implementations (see page 351) for more details.

2. Add your Design Files and Set Implementation Options

In the Projects View, click on the "quickstart" project to select it. Now click on the **Add Files** ( ) toolbar button. In the Add Source Files Dialog (see page 261), select `quickstart.vma`, `quickstart.pdc`, and `quickstart.sdc` by holding down the **CTRL** key and clicking on them. 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 355) or Working with Projects and Implementations (see page 351) for more details.

3. Run the Flow

In the Flow View (see page 187), click on the **Run Flow** ( ) toolbar button. Output from the Flow (see page 316) will be shown in the Tcl Console View (see page 258). When the flow is finished running, you will see the Flow Steps (see page 317) in the Flow View updated with a green check mark ( ) to indicate success, and all newly generated reports will be displayed in the editor area.

See the Flow (see page 316) concept or Running the Flow (see page 361) 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 174) to analyze critical paths and highlight them in the Floorplanner View (see page 178). Clicking the **Zoom To Path** ( ) toolbar button in the Critical Paths View will zoom the Floorplanner View (see page 178) to the path currently selected in the Critical Paths View. Use the Search View (see page 245) and Selection View (see page 249) to locate objects of interest. Clicking the **Zoom To Selection** ( ) toolbar button in the Selection View will zoom 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 23). Perspectives contain views (see page 163) and editors (see page 25) 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 a user must view 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 163), editors (see page 25), menus, and toolbars in the Workbench (see page 23) window.

For example, the Projects perspective combines views (see page 163) commonly used while managing project source files, while the Floorplanner perspective contains the views that are used while viewing chip layout and floorplanning information. Users frequently switch perspectives while working inside the Workbench (see page 23).

> Within the Workbench (see page 23) window, all perspectives share the same set of Editors (see page 25). All editors are usable/visible from all perspectives. Likewise, each of the Views (see page 163) may optionally be used within any perspective, but they're most useful when grouped with the other views from their native perspective. One of the views, the Tcl Console View (see page 258), is a member of all the perspectives.

Projects Perspective

The ( )Projects Perspective allows the user to select an active project and implementation, manage the contents and configuration of the active project/implementation, run the Flow (see page 316), and view the reports generated by the Flow.

By default, this perspective contains the Projects View (see page 238), Flow View (see page 187), Options View (see page 215), TCL Console View (see page 258), and the Editor area, which can contain any ACE Editor or Report. The Multiprocess View (see page 204) is also part of this perspective, but is hidden by default.

For more information, see Working with Projects (see page 351), Running the Flow (see page 361), and Using the Tcl Console (see page 382).

Floorplanner Perspective

The ( )Floorplanner Perspective allows the user to view and edit the placement and routing of their active project /implementation.

By default, this perspective contains the Floorplanner View (see page 178), Package View (see page 226), Search View (see page 245), Selection View (see page 249), Critical Paths View (see page 174), Critical Path Diagram View (see page 170), IO Assignment View (see page 191), Netlist Browser View (see page 211), Clock Domains View (see page 165), Clock Regions View (see page 167), Placement Regions View (see page 234), Partitions View (see page 231), and TCL Console View (see page 258).

For more information on using the views in this perspective, see Viewing the Floorplanner (see page 392), Viewing the Package Layout (see page 498), Pre-Placing a Design (see page 399), Analyzing Critical Paths (see page 404), and Managing I/Os (see page 501).

> Unlike all other perspectives, the Floorplanner perspective hides the Editor area. To view editors (see page 25) and reports (see page 322), 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 Projects View (see page 238), IP Libraries View (see page 197), IP Diagram View (see page 196), IP Problems View (see page 198), Outline View (see page 225), TCL Console View (see page 258) , and the Editor Area, which can contain any ACE Editor or Report.

See Creating an IP Configuration (see page 387) for more details.

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 Snapshot Debugger View (see page 253), Download View (see page 176), TCL Console View (see page 258), JTAG Browser View (see page 199), JTAG Diagram View (see page 203), and 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 415) and Playing a STAPL File (Programming a Device) (see page 432)

HW Demo Perspective

The ( )HW Demo Perspective allows the user to observe various aspects of a particular device, by selecting one of the provided demonstration designs from a list. Once 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 Snapshot Debugger View (see page 253), HW Demo View (see page 189), Tcl Console View (see page 258), and 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 441).

Editors

Most Perspectives (see page 23) in the Workbench (see page 23) are comprised of an editor area and one or more views (see page 163). 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 238), 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 will provide 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 347) 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 322) generated by ACE. By default, when ACE is running the Flow (see page 316) in single-process mode, ACE will open HTML versions of the reports in the HTML Report Browser (see page 25) as soon as the report data is generated/updated. When ACE is in Multiprocess mode (via the Multiprocess View (see page 204)), only the Multiprocess Summary Report (see page 323) is automatically opened in the editor area – the other reports must be opened manually through the Projects View (see page 238), or by following the Timing Report hyperlinks for each Implementation (see page 312) 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 387).

> 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 IP Configuration Editors available at the customer site may only be a subset of what is documented in this user guide. For example, if the customer's installed and licensed devices include no LRAMs, then any IP tools based upon LRAMs will not be available within the customer's ACE GUI.

HTML Report Browser

When HTML versions of generated Reports (see page 322) 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.

> 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.

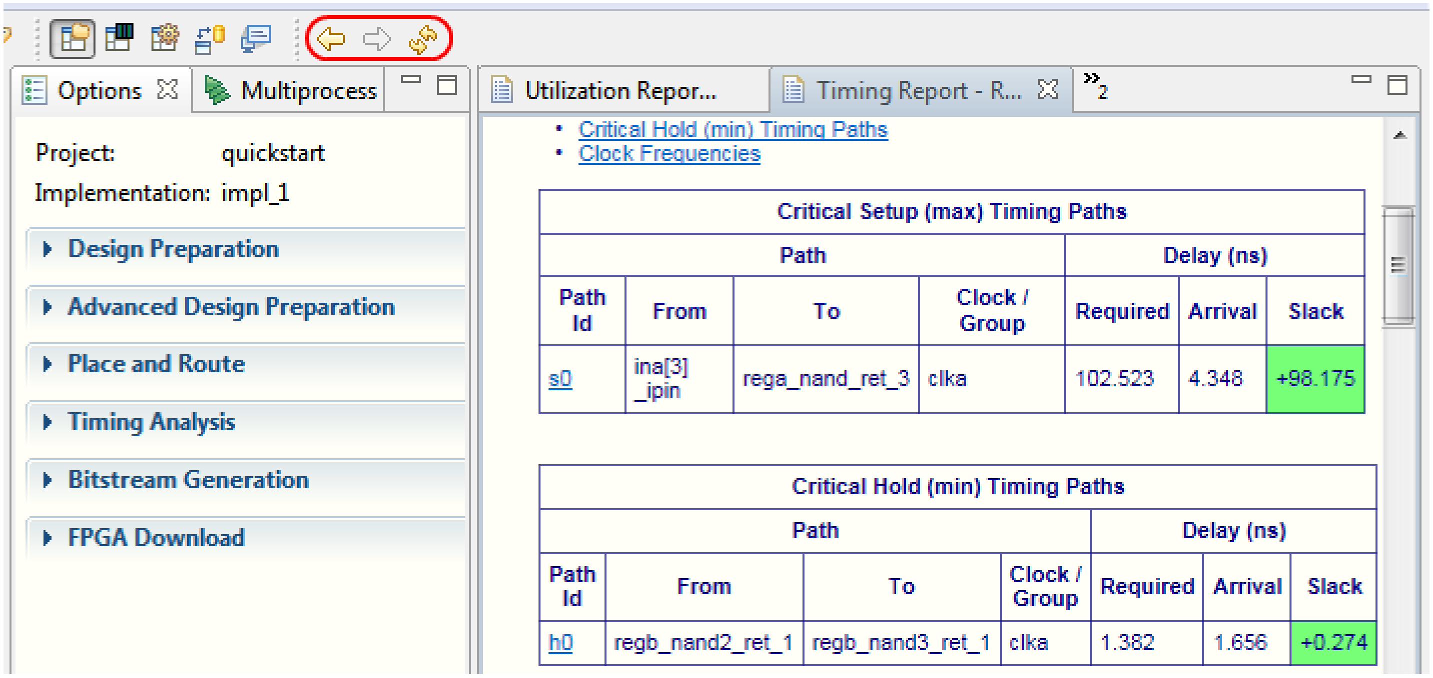


Figure 1: ***HTML Report Browser Screenshot, with Toolbar Button Locations Circled in Red***

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.

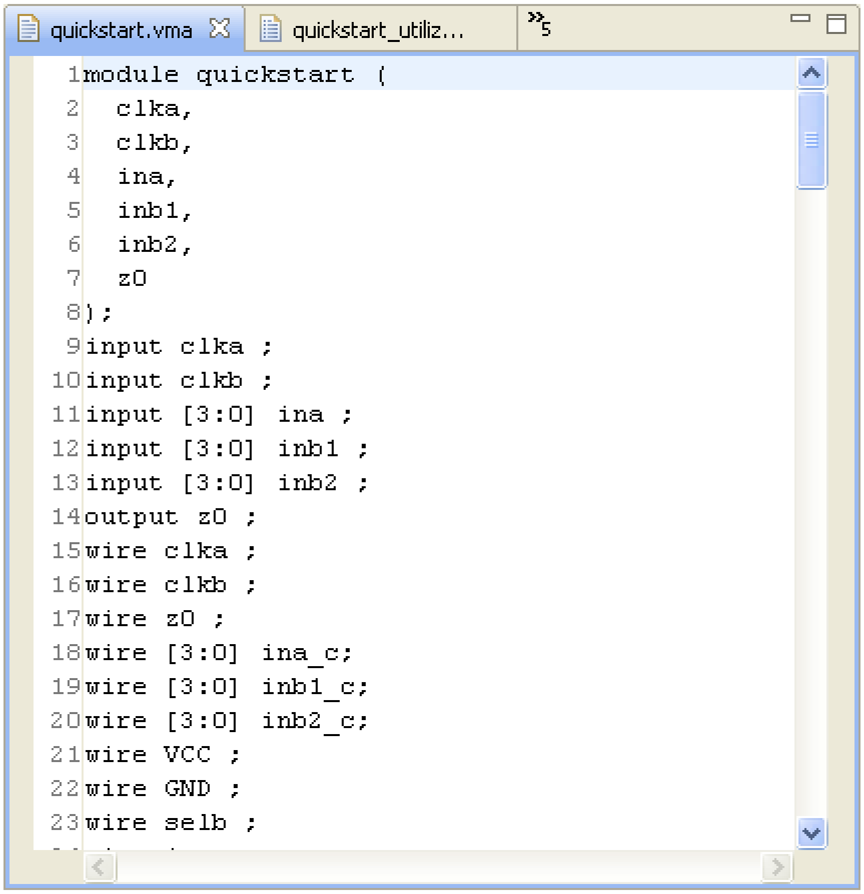


Figure 2: Screenshot of the default Text Editor

VCD Waveform Editor

The VCD Waveform Editor does not allow the user to edit 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 25), and it opens whenever the user selects a VCD file, we'll think of it as an editor in read-only mode.

The waveform viewer allows the user to examine 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 415). (See also: Snapshot Debugger View (see page 253))

As with familiar waveform editors, the user will be able to manipulate the placement of a Marker (here, a pink vertical line) with the mouse in the graphical waveform area, so that they can see the value of all signals at the same instant of time. The user will also be able to re-order the signals (move an individual signal vertically amongst its peers) and hide/show individual signals. 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 will remember signal name ordering, panel sizes, zoom level, and the sample offset between file loads. These will be remembered between Snapshot captures within a single session, as well as between ACE sessions.

> **Note**
>
> In addition to the graphical waveform view, the user may view the raw text content of the VCD file. 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.

> **Note**
>
> None of the actions available in the VCD Waveform Editor will change the content stored in the VCD file.

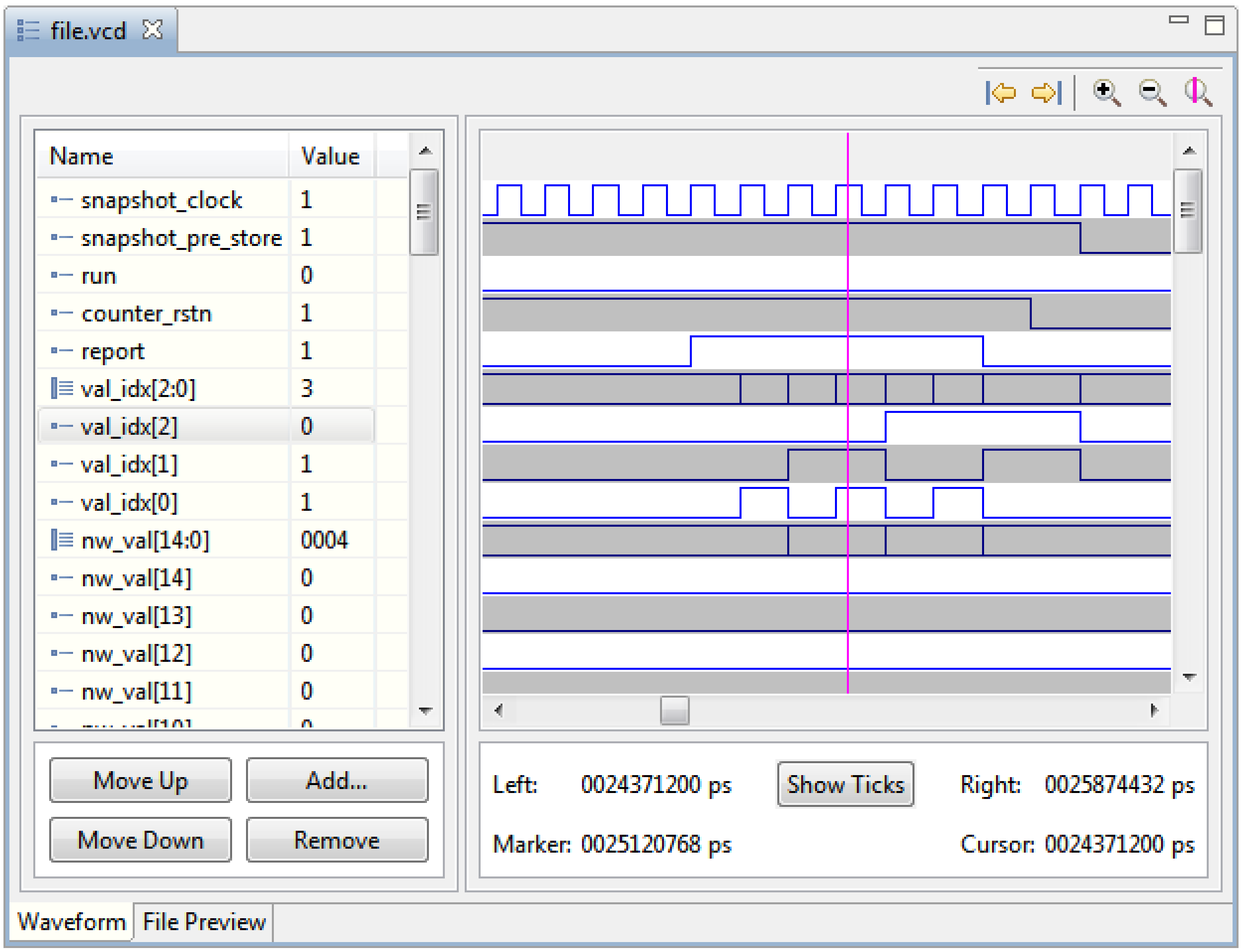


Table 2: *VCD Waveform Editor Options*

Option	Description
Signal Value Table	
Signal 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.

Option	Description
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 will 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 259), 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.

Icon	Action	Description
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.
	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).

Speedster22i Advanced PLL Configuration Editor

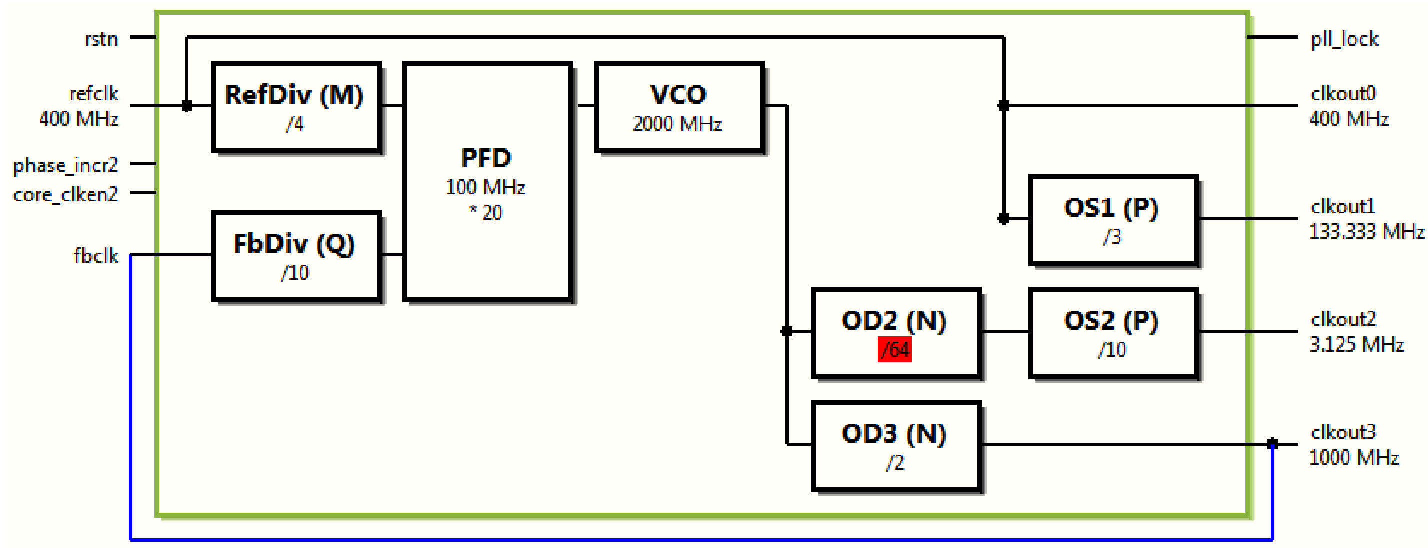
The Advanced PLL Configuration Editor provides a graphical wizard for creating a PLL configuration file (.acxip). This editor allows the user to generate the required configuration files for design with the embedded PLLs. See Creating an IP Configuration (see page 387). Unlike the much simpler Speedster22i Basic PLL Configuration Editor (see page 43), the Advanced PLL Configuration Editor allows the user to access the complete functionality of the PLL.

By default, the Advanced PLL Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**). The Advanced PLL configuration information is broken up into several pages, organized by concept. While some pages are always available, many pages of configuration options may appear or disappear depending upon whether they are relevant, based upon the user's current choices for configuration options.

Once the user has configured the PLL to meet their requirements, and the Advanced PLL Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The IP Diagram View (see page 196) for the PLL shows live information about the current configuration in the Editor, including which logic blocks are currently active, which inputs and outputs are currently active, and what the various frequencies are within the PLL. Additionally, configuration errors will be shown with a red background, and configuration warnings will be shown with a yellow background (these are the default IP Diagram colors, and may be modified in the Preferences).



1. Overview Page (see page 32)
2. Placement Page (see page 35)
3. Clock Output [0,1,2,3] Pages (see page 36)
4. Output Divider [0,1,2,3] Pages (see page 37)
5. Output Synthesizer [0,1,2,3] Pages (see page 40)
6. Port Names Page (see page 40)

Advanced PLL Overview Page

The Overview page contains the top-level, global properties that govern the structure and base configuration of the PLL. Changes made on this page affect all the outputs for this PLL.

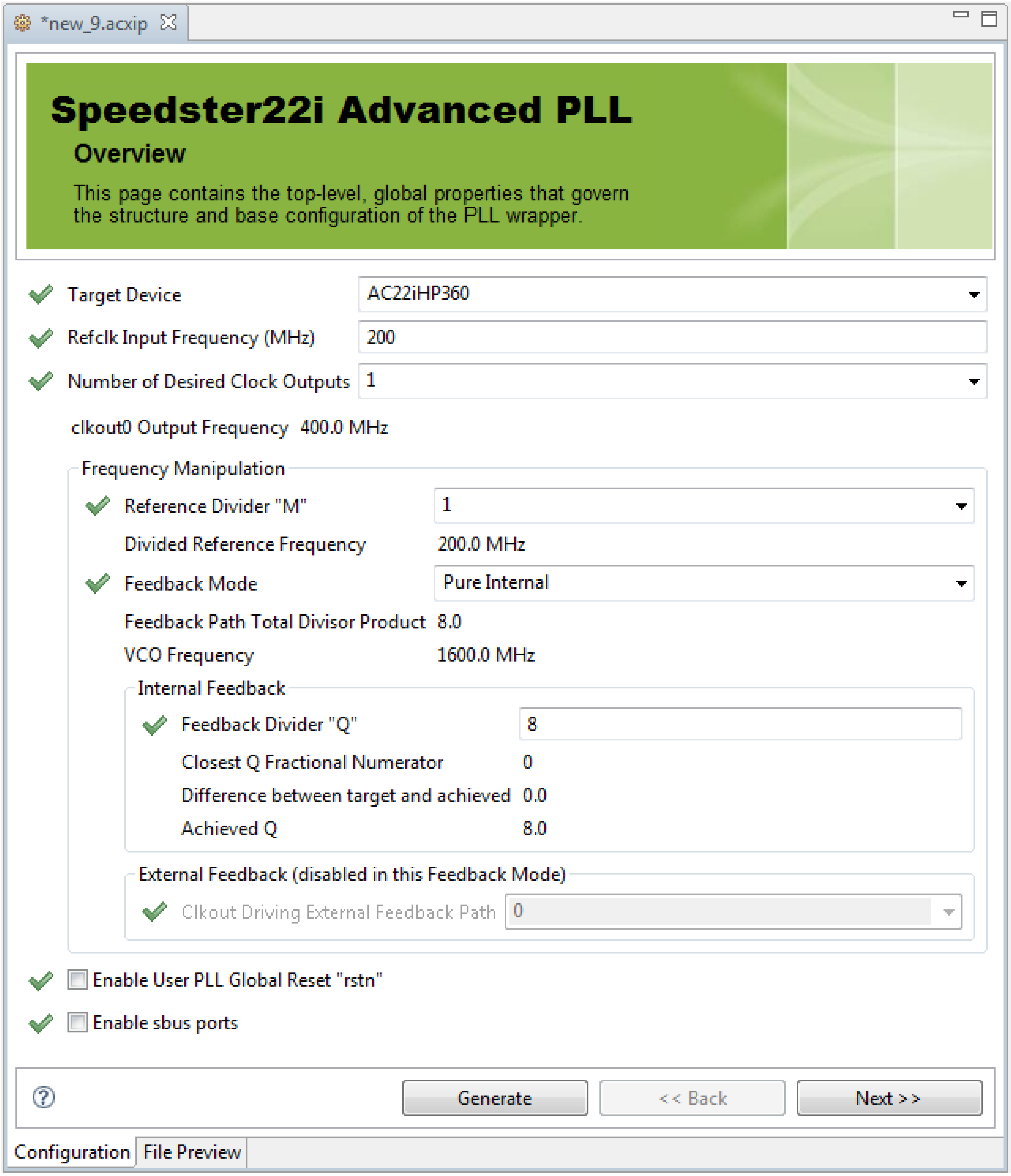


Table 5: ***PLL Editor Overview Page Options***

Option	Editable	Description
Target Device	Y	The Speedster22i device this PLL is intended to target. **Ensure 'Target Device' is set correctly** Many vital details of PLL functionality can differ by device. If set incorrectly, the instantiated PLL may not work.
Refclk Input Frequency (MHz)	Y	The frequency of the PLL reference clock input.
Number of Desired Clock Outputs	Y	The number of desired clock output signals for this PLL. Changing this will alter the number of active pages of Clock Output configuration options.
clkout Output Frequency		The calculated output frequency of the named clock output signal. The number of outputs listed will match the "Number of Desired Clock Outputs".
Reference Divider "M"	Y	The amount by which the reference clock frequency should be divided before entering the PLL's Phase Frequency Detector. As this value increases, the "VCO Frequency" decreases.
Divided Reference Frequency		The calculated reference clock frequency after having been divided by the "Reference Divider 'M'".
Feedback Mode	Y	Selects one of the three allowed feedback modes, and enables/disables related options on this page according to the selected mode. See Feedback Modes below for more details.
Feedback Path Total Divisor Product		The calculated total product of all divisors in the present feedback path.
VCO Frequency		The calculated VCO output frequency. The algorithm used will vary depending upon the selected Feedback Mode.
Feedback Divider "Q"	Y	The amount by which the feedback signal should be divided before entering the PLL's Phase Frequency Detector. As this value increases, the "VCO Frequency" increases. When in Pure Internal Feedback Mode, this may be a floating-point value; in Mixed feedback mode this must be an integer value. In Pure External Feedback Mode, this option will be disabled.

Option	Editable	Description
Closest Q Fractional Numerator		This calculated values represents the fractional portion of the entered "Feedback Divider 'Q'" used to configure the PLL. When floating point values are entered for the "Feedback Divider 'Q'", they must be represented as fractions of 65536 (a 16-bit representation of the fraction is used). The value displayed provides the closest possible fraction to that requested by the user.
Difference between target and achieved		This calculated value shows how close the PLL can come to the requested "Feedback Divider 'Q'" value. (Some requested fractional values are impossible to exactly represent within the 16 bits available in the PLL.)
Achieved Q		This calculated value shows the exact "Feedback Divider 'Q'" value which will be used by the PLL. This may differ slightly from the requested value; the closest approximate value will be chosen if an exact match is not possible.
Clkout Driving External Feedback Path	Y	Selects which of the currently-enabled clock outputs will drive the external feedback path. Note that Phase Adjustment is not allowed in the Feedback Path.
Enable User PLL Global Reset "rstn"	Y	If enabled, the input port "rstn" may be used to dynamically trigger a global reset of the PLL.
Enable sbus ports	Y	When selected, the sbus ports will be exposed in the generated HDL file(s).

Feedback Modes

There are three feedback modes available to the PLL. Some allow fractional feedback, others allow deskew.

Phase Adjustment is not allowed in Pure External Feedback or Mixed Feedback paths

Feedback using a clock output that has Phase Adjustment enabled is disallowed. Phase rotation in the feedback path will cause the PLL to unlock. If Phase Adjustment is required on any output clocks, it must be used on outputs which are not in the feedback path.

Table 6: ***PLL Feedback Modes***

Mode	Fractional Feedback?	Deskew?	Algorithm
Pure Internal	Y	N	$F_{VCO} = (Q / M) * F_{ref}$
Pure External	N	Y	$F_{VCO} = (N * P / M) * F_{ref}$
Mixed	N	Y	$F_{VCO} = (Q * N * P / M) * F_{ref}$

Pure Internal Feedback

When internal feedback mode is selected, the VCO clock is divided by the "Feedback Divider 'Q'" only. In this mode, the PLL can have both integer and fractional feedback divider values. The "Clkout Driving External Feedback Path" option becomes disabled in this mode, since it is irrelevant. Deskewing is not possible in this mode.

Pure External Feedback

When external feedback mode is selected, the VCO clock is divided by the Output Divider (N) and (optionally) the Output Synthesizer (P) of one of the clkouts. The "Clkout Driving External Feedback Path" option becomes enabled in this mode, and the "Feedback Divider 'Q'" option becomes disabled. In this mode, feedback using a clock output that has Phase Adjustment enabled is disallowed.

Mixed Feedback

Mixed feedback mode is a modified version of Pure External feedback mode, in that the Feedback Divider integer value is also in the deskew feedback path. As with Pure External mode, Phase Adjustment is not allowed in the feedback path. The "Clkout Driving External Feedback Path" option becomes enabled in this mode, as does "Feedback Divider 'Q'".

Placement Page

The Placement page contains configuration information relating to the PLL's placement in the Speedster device. The site chosen will be exported in a placement constraints (`.pdc`) file when the user chooses to generate their PLL design files.

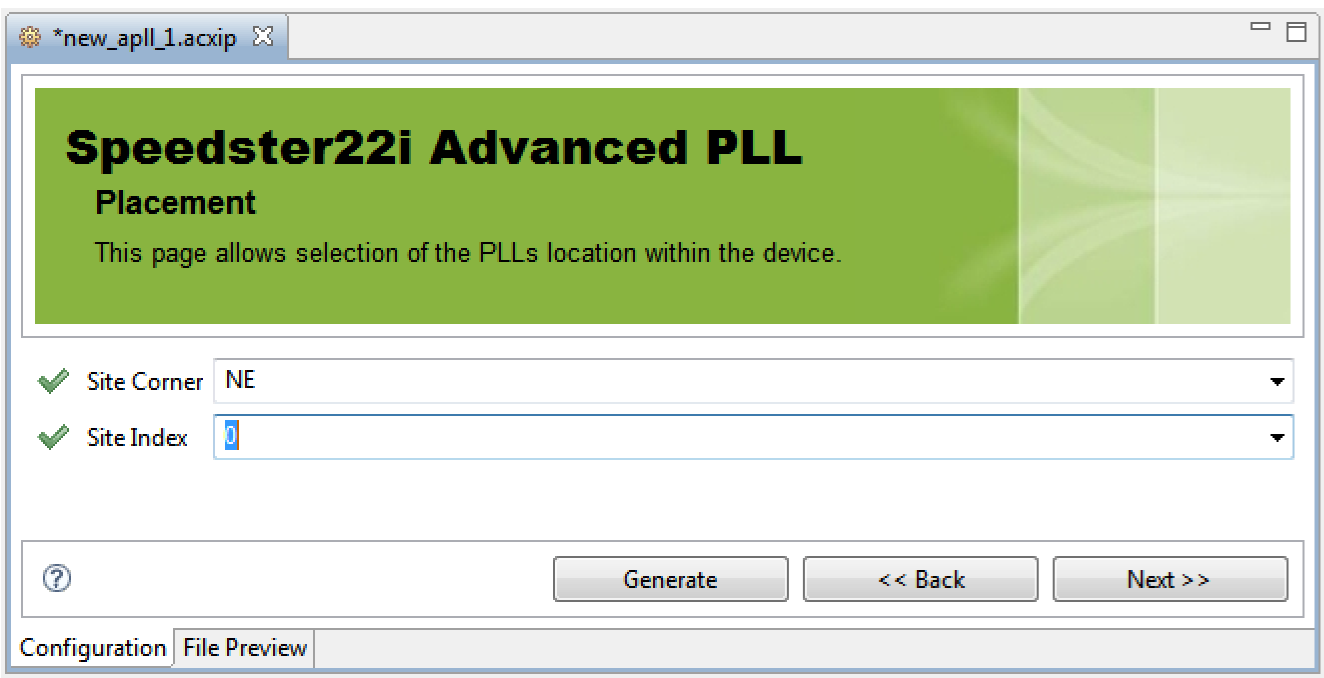


Table 7: ***PLL Editor Placement Page Options***

Option	Editable	Description
Site Corner	Y	The corner of the Speedster22i device where this PLL instance should be placed. The four choices are **NE** , **SE** , **SW** , and **NW** .

Option	Editable	Description
Site Index	Y	The site index within the corner where this PLL instance should be placed.

Clock Output 0 - 3 Pages

The Clock Output pages each contain general configuration information relating to a single PLL output signal. Since there are one-to-four PLL output signals per PLL (as configured on the Overview Page (see page 32)), between one and four Clock Output pages will be enabled.

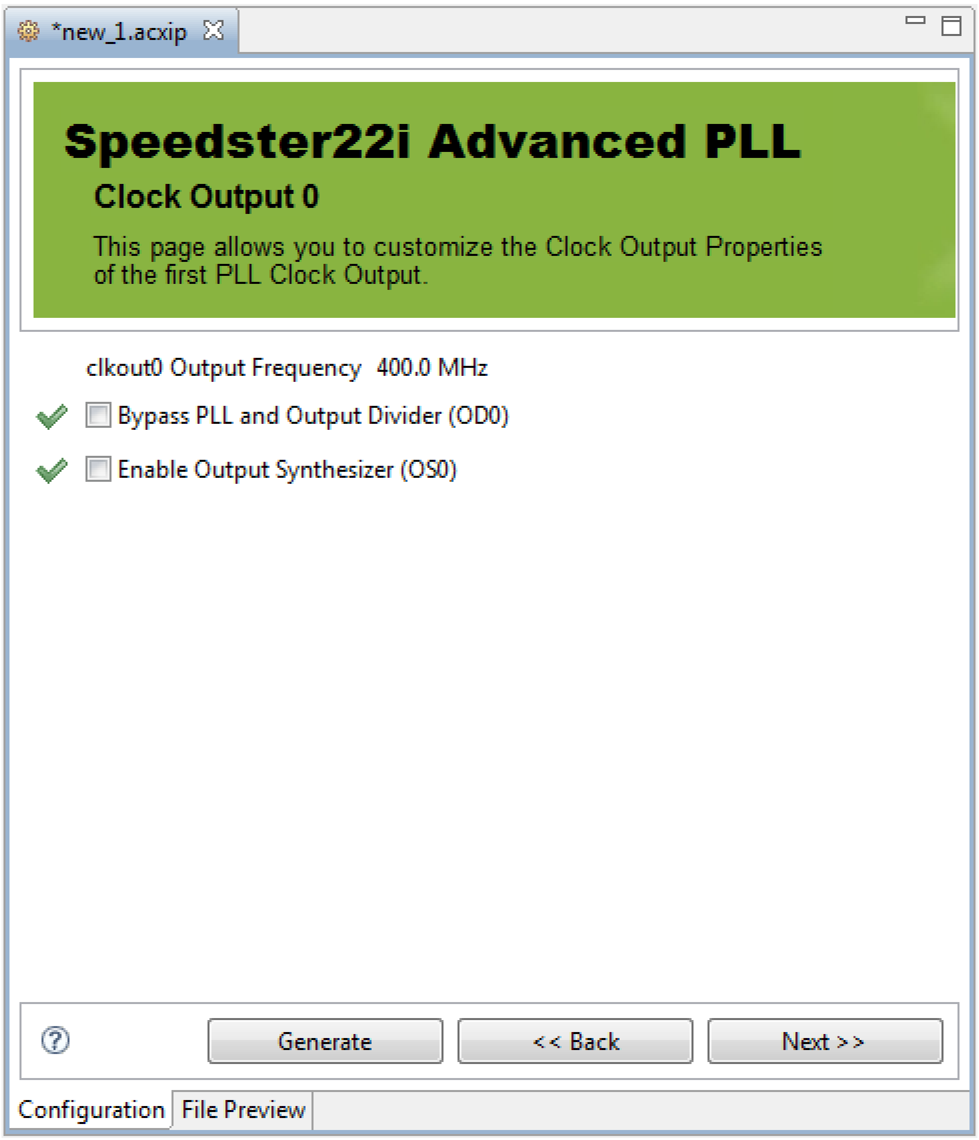


Table 8: ***PLL Editor Clock Output Page Options***

Option	Editable	Description
clkout Output Frequency		The calculated frequency of this clock output signal as it exits the PLL.
Bypass PLL and Output Divider (OD [0-3])	Y	Enabling this will bypass the Reference Divider, PFD, VCO, and OD, sending the reference clock input signal directly to the OS input. Disabling this allows normal PLL behavior. Note that when this is enabled, the IP Editor configuration page for the Output Divider (OD) associated with this clock output is disabled. In addition, when enabled, it becomes illegal to use this output in an external feedback path, as the PLL will lose lock.
Enable Output Synthesizer (OS[0-3])	Y	Allows this clock output to use its Output Synthesizer (OS). When enabled, this activates the IP Editor configuration page for the OS associated with this clock output. When disabled, the associated OS configuration page is hidden.

Output Divider 0 - 3 Pages

The Output Divider pages contain configuration information relating to the output divider (OD) for a single PLL clock output signal. Since there are one-to-four PLL output signals per PLL, there are also up to four of these configuration pages. Because the PLL and OD logic may be bypassed on a per-output basis (as configured on the Clock Output [0,1,2,3] Pages (see page 36)), this page may sometimes be hidden, even if the related clock output is enabled.

> When enabled, Phase Adjustment is applied according to the frequency of the input to the OD block, before the OD divider is applied. Thus, it will shift the incoming VCO high-frequency signal by increments of 45 degrees. Higher frequency VCO output allows smaller granularity phase shifting (in terms of ps).

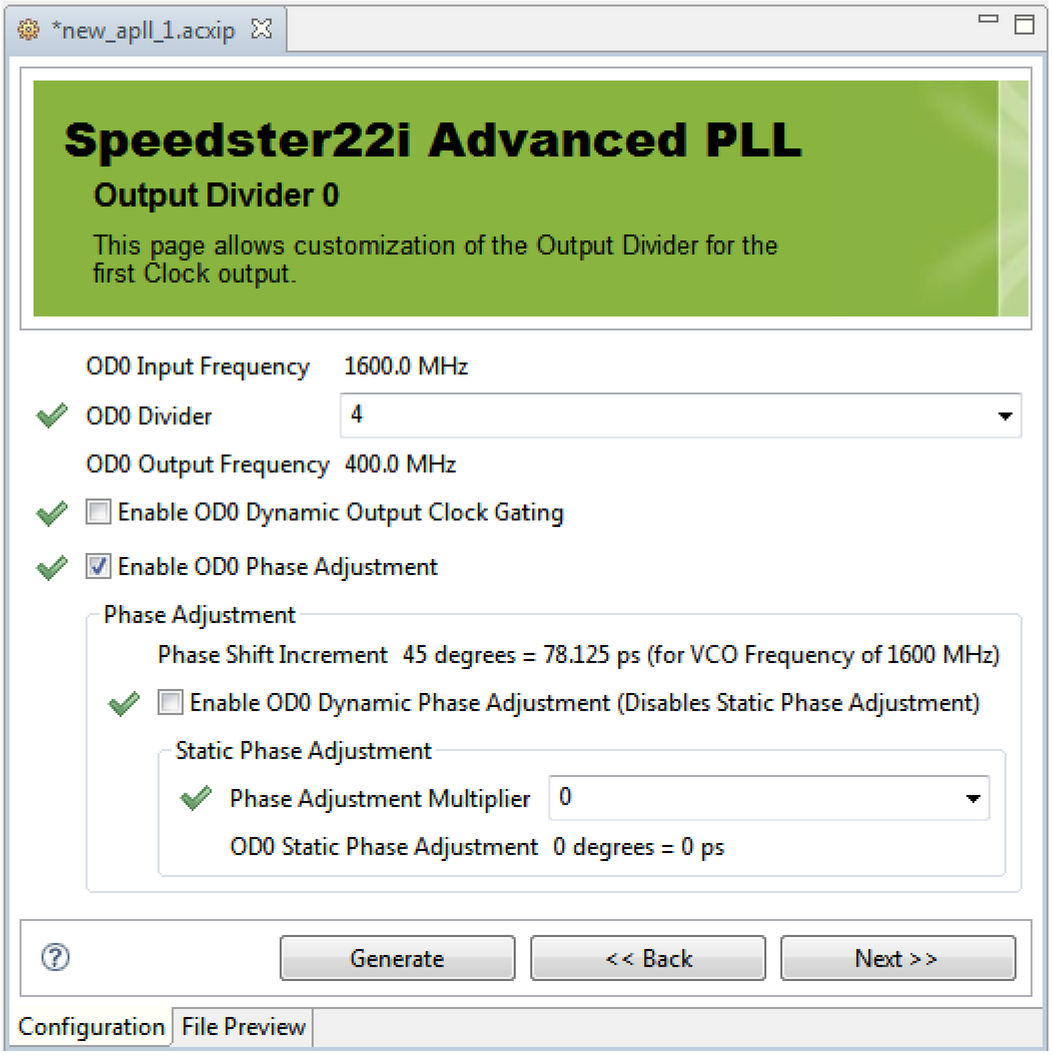


Figure 3: Output Divider 0 screenshot with Static Phase Adjustment enabled

Table 9: *PLL Editor Output Divider Page Options*

Option	Editable	Description
OD[0-3] Input Frequency		The calculated frequency of the input signal as it enters this OD logic block.
OD[0-3] Divider	Y	The factor by which the signal entering the OD should be divided before it exits the OD. As this increases, the OD output frequency decreases. (Unless this OD is in the external feedback path, in which case increasing this value does not affect the OD output frequency, but increases the VCO output frequency.)
OD[0-3] Output Frequency		The calculated frequency of the output signal as it exits this OD logic block.
Enable OD[0-3] Dynamic Output Clock Gating	Y	Enabling this will allow the OD output clock to be gated based upon the value of the corresponding input port `core_clken`
Enable OD[0-3] Phase Adjustment	Y	Enabling this will allow the OD to perform phase adjustment on its input signal (from the VCO).
Phase Shift Increment		The incremental step by which the phase will shift for each phase_incr pulse, based upon 1/8 the VCO output frequency's period; calculated as ((1000000 / VCO frequency) / 8)
Enable OD[0-3] Dynamic Phase Adjustment (Disables Static Phase Adjustment)	Y	Enabling this selects dynamic phase adjustment mode, where the OD input signal's phase is incremented by 45.0 degrees every time a rising edge is applied to the corresponding `phase_incr` input port. When disabled, Static Phase Adjustment mode is used instead, where the input signals's phase is adjusted a by a fixed number of degrees (as configured by the Phase Adjustment Multiplier).
Phase Adjustment Multiplier	Y	Selects how far the OD input phase should be shifted from the VCO phase (this * 45.0 degrees = total phase shift).
OD[0-3] Static Phase Adjustment		The amount (reported in both degrees and ps) the OD input will be shifted relative to the VCO output.

Output Synthesizer 0 - 3 Pages

The Output Synthesizer (OS) pages contain configuration information relating to the OS associated with the PLL's selected clock output. Because the OS can be disabled for a PLL clock output, this page is only visible when the "Enable Output Synthesizer (OS [0-3])" field on the appropriate Clock Output [0,1,2,3] Page (see page 36) is enabled.

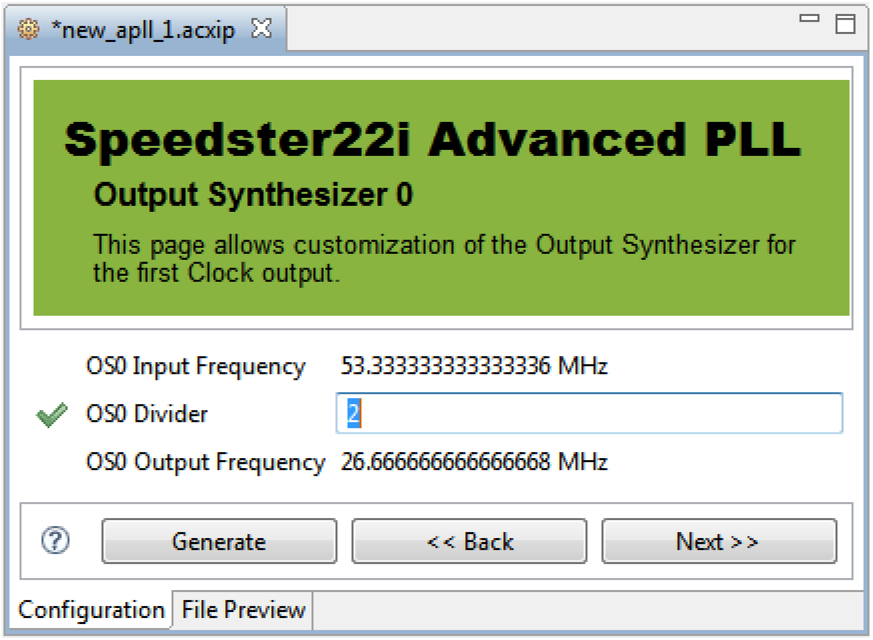
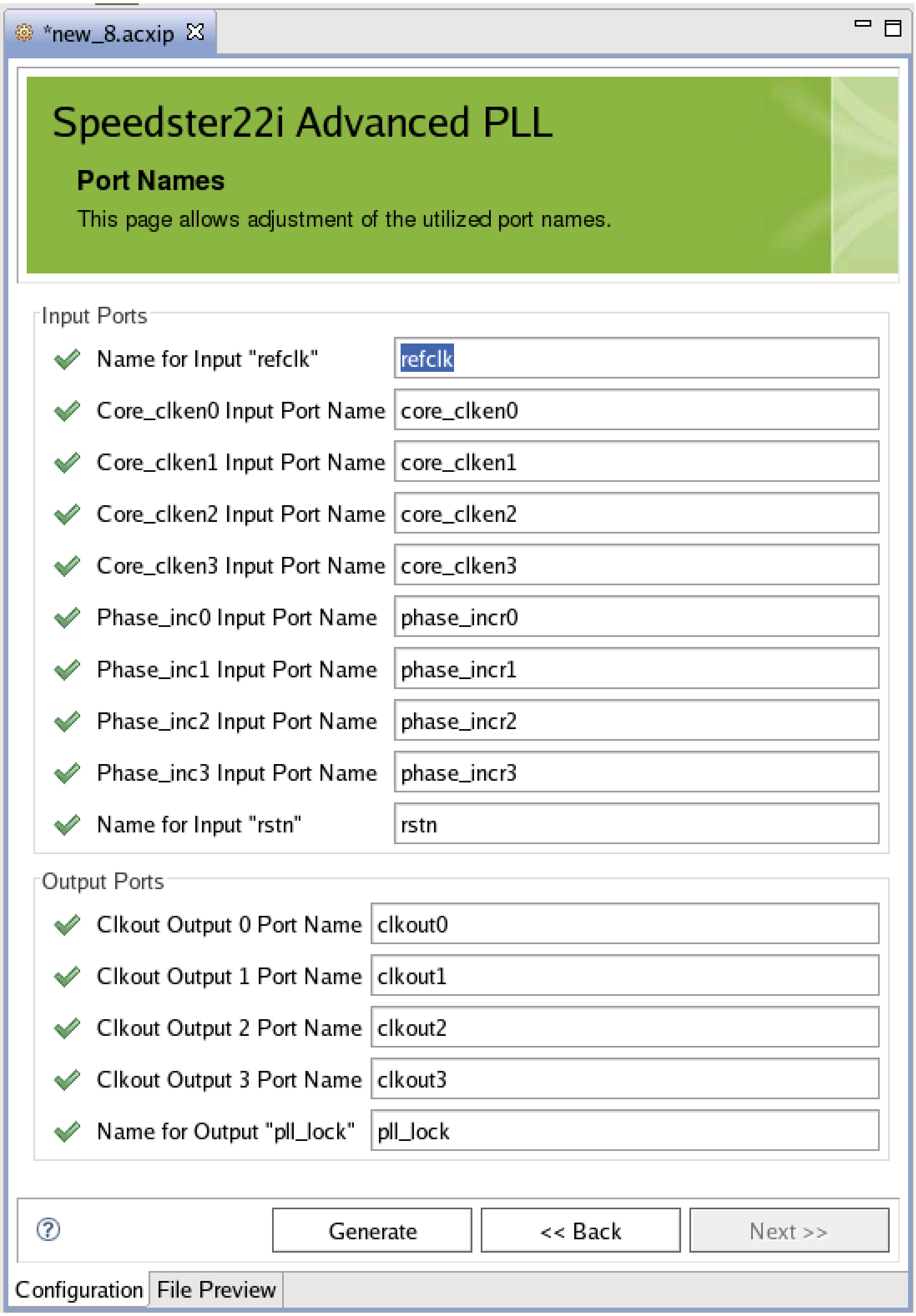


Table 10: *PLL Editor Output Synthesizer Page Options*

Option	Editable	Description
OS [0-3] Input Frequency		The calculated frequency of the OS input signal.
OS [0-3] Divider	Y	The factor by which the signal entering the OS should be divided before it exits the OS . As this increases, the OS output frequency decreases. (Unless this OS is in the external feedback path, in which case increasing this value does not affect the OS output frequency, but instead increases the VCO output frequency.)
OS [0-3] Output Frequency		The calculated frequency of the OS output signal.

Port Names Page

The Port Names page contains all the input and output ports which will be used by the PLL in its current configuration. (Changing options on other pages will show and hide port names on this page, as the need for the ports changes.)



All port names entered on this page must adhere to Verilog *and* VHDL naming standards. Illegal names will be caught as errors, and will prohibit RTL wrapper file generation.

Table 11: ***Advanced PLL Editor Port Names Page Options***

Option	Description
Input Ports	
Name for Input "refclk"	The desired name for the reference clock input signal in the generated RTL.
Name for Input "fbclk"	The desired name for the feedback clock input signal in the generated RTL. This option is not available when the PLL is in Pure Internal Feedback Mode.
Core_clken [0-3] Input Port Name	The desired name for the core_clken input signal (for the appropriately numbered clock output) in the generated RTL. This port is not available unless the associated clock output is enabled, and dynamic clock gating is enabled for said clock output.
Phase_inc[0-3] Input Port Name	The desired name for the phase_inc input signal (for the appropriately numbered clock output) in the generated RTL.
Name for Input “rstn”	The desired name for the rstn input signal in the generated RTL. This port is not available unless “Enable User Global PLL Reset” is selected on the Overview page.
Output Ports	
Clkout Output [0-3] Port Name	The desired name for the PLL clock output in the generated RTL.
Name for Output "pll_lock"	The desired name for the PLL's lock indication output signal in the generated RTL.

Speedster22i Basic PLL Configuration Editor

The Basic PLL Configuration Editor provides a simplified graphical wizard for creating a PLL configuration file (`.acxip`). This editor allows the user to generate the required configuration files for design with the embedded PLLs. See Creating an IP Configuration (see page 387). Unlike the much more complicated Speedster22i Advanced PLL Configuration Editor (see page 30), the Basic PLL Configuration Editor allows the user to access only the most-often used functionality of the PLL.

By default, the Basic PLL Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**). The Basic PLL configuration information fits into a single page.

Once the user has configured the PLL to meet their requirements, and the Basic PLL Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

> The Basic PLL Editor only supports **Pure Internal Feedback Mode**, and thus hides the choice of feedback path modes from the user. This allows fractional feedback dividers (so any desired `clkout0` output frequency is achievable), but prohibits deskew functionality.
>
> For simplification, output phase adjustment is also disabled.
>
> If deskew or phase adjustment functionality are required, the Advanced PLL Configuration Editor must be used instead of the Basic PLL Editor.

IP Diagram

The IP Diagram View (see page 196) for the Basic PLL shows live information about the current configuration in the Editor, including which logic blocks are currently active, which inputs and outputs are currently active, and what the various frequencies are within the PLL. Additionally, configuration errors will be shown with a red background, and configuration warnings will be shown with a yellow background (these are the default IP Diagram colors, and may be modified in the Preferences).

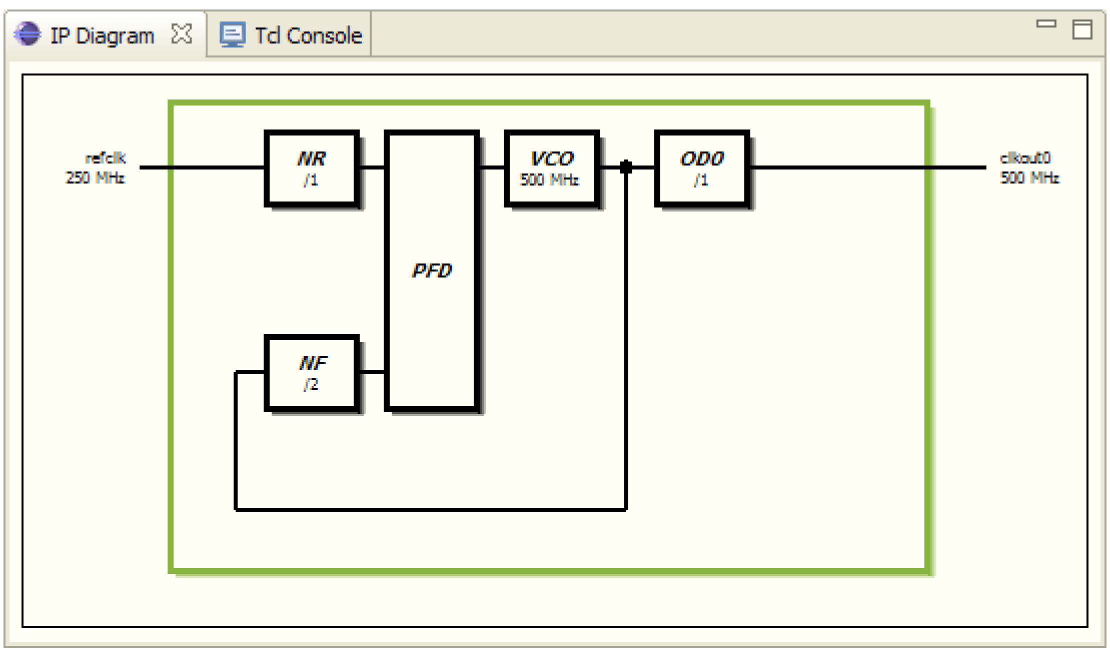


Figure 4: Screenshot showing the Basic PLL IP Diagram

Basic PLL Overview Page

The Overview page contains all the properties that govern the structure and configuration of the basic PLL.

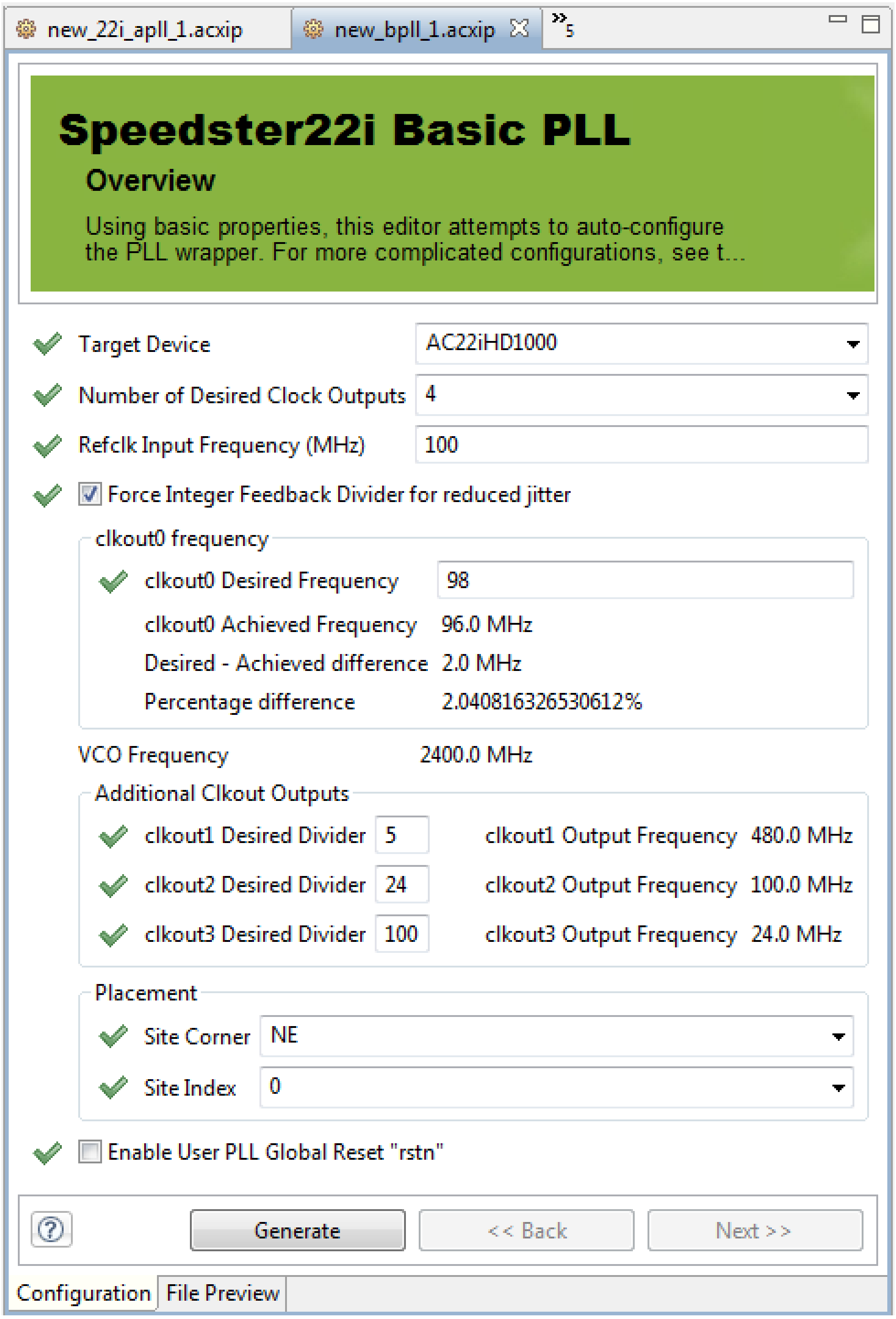


Table 12: *PLL Editor Overview Page Options*

Option	Editable	Description
Target Device	Y	The Speedster22i device this PLL is intended to target. **Ensure Target Device is Set Correctly** Many vital details of PLL functionality can differ by device. If set incorrectly, the instantiated PLL may not work.
Number of Desired Clock Outputs	Y	The number of desired clock output signals for this PLL.
Refclk Input Frequency (MHz)	Y	The frequency of the PLL reference clock input.
Force Integer Feedback Divider for reduced jitter	Y	Feedback dividers in integer mode have reduced jitter, but have a coarser granularity, and thus may not be able to achieve the exact desired clkout0 output frequency. In integer mode, users may need to alter the **Refclk Input Frequency** to exactly achieve the **clkout0 Desired Frequency**.
clkout0 frequency		
clkout0 Desired Frequency	Y	The frequency desired for clkout0. ACE will automatically choose PLL configuration values (M, Q.R, N, P) to get as close to the desired frequency as possible.
clkout0 Achieved Frequency		The calculated output frequency of the clkout0 clock output signal. This will be as close to the "clkout0 Desired Frequency" as possible.
Desired - Achieved Difference		The difference between the **clkout0 Desired Frequency** and the **clkout0 Achieved Frequency** in MHz.
Percentage Difference		The percentage mismatch between the **clkout0 Desired Frequency** and the **clkout0 Achieved Frequency**
VCO Frequency		The calculated VCO output frequency which was required to achieve the requested clkout0 frequency. This will also be used to drive any other enabled clkout outputs.
Additional Clkout Outputs		
clkout[1-3] Desired Divider	Y	The requested divider to use to alter the VCO Frequency for the clkout output. Some values in the allowed range are not achievable - in these cases, a warning is reported and the closest divider is used instead. Changing this option value will change the associated clkout Output Frequency.

Option	Editable	Description
clkout[1-3] Output Frequency		The achieved output frequency of the named output. This will vary according to the VCO Frequency and the clkout Desired Divider.
Placement		
Site Corner	Y	The corner of the Speedster22i device where this PLL instance should be placed. The four choices are **NE**, **SE**, **SW**, and **NW**.
Site Index	Y	The site index within the corner where this PLL instance should be placed.
Enable User PLL Global Reset "rstn"	Y	If enabled, the input port "rstn" may be used to dynamically trigger a global reset of the PLL.

Speedster22i BRAM Configuration Editor

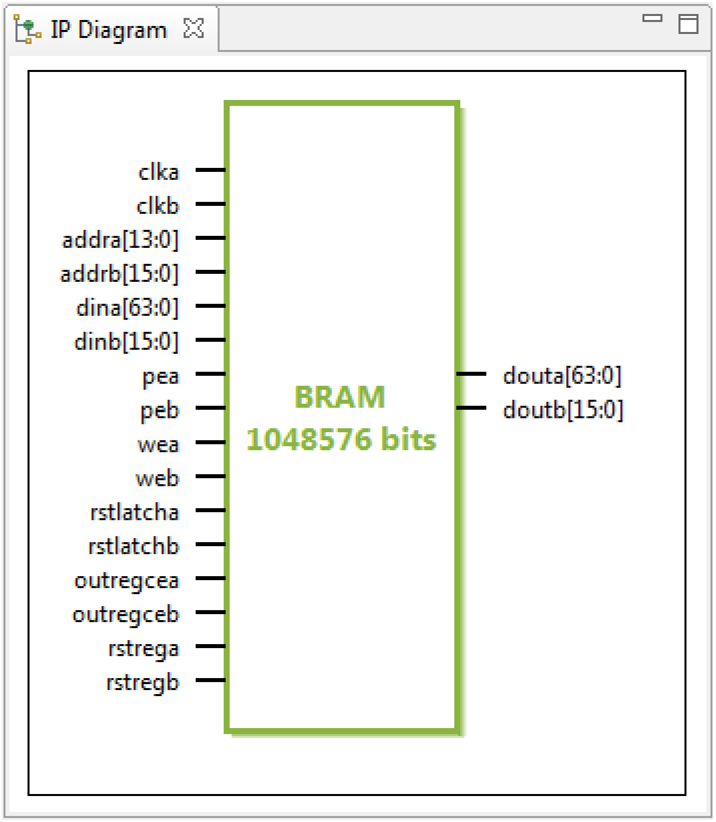
The BRAM Configuration Editor provides a simplified graphical wizard for creating a BRAM configuration file (.acxip). This editor allows the user to generate the required design files for BRAM configuration. See Creating an IP Configuration (see page 387).

By default, the BRAM Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**). The BRAM configuration information fits into a single page, unlike more complicated IP editors.

Once the user has configured the BRAM to meet their requirements, and the BRAM Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The IP Diagram View for the BRAM shows live information about the current configuration in the Editor, including which inputs and outputs are currently active.



1. Overview Page (see page 47)

BRAM Overview Page

The Overview page contains all the properties that govern the structure and configuration of the BRAM wrapper.

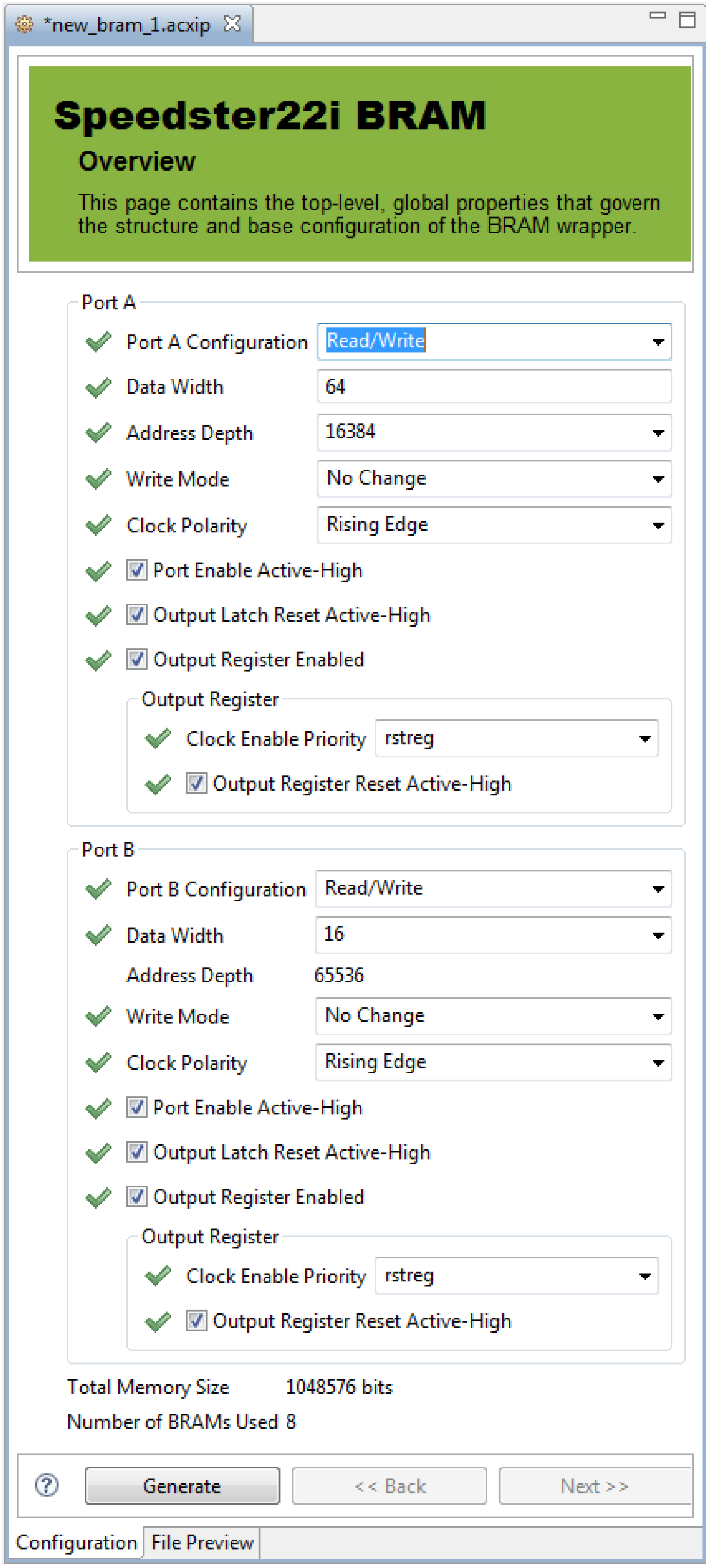

*new_bram_1.acxip
Speedster22i BRAM
Overview
This page contains the top-level, global properties that govern the structure and base configuration of the BRAM wrapper.
Port A
Port A Configuration
Read/Write
Data Width
64
Address Depth
16384
Write Mode
No Change
Clock Polarity
Rising Edge
Port Enable Active-High
Output Latch Reset Active-High
Output Register Enabled
Output Register
Clock Enable Priority
rstreg
Output Register Reset Active-High
Port B
Port B Configuration
Read/Write
Data Width
16
Address Depth
65536
Write Mode
No Change
Clock Polarity
Rising Edge
Port Enable Active-High
Output Latch Reset Active-High
Output Register Enabled
Output Register
Clock Enable Priority
rstreg
Output Register Reset Active-High
Total Memory Size
1048576 bits
Number of BRAMs Used 8
Generate
<< Back
Next >>
Configuration
File Preview

Table 13: *BRAM Editor Overview Page Options*

Option	Editable	Description
Port A		
Port A Configuration	Y	BRAMs can be configured for read, write, or read/write capability independently on both Port A and Port B sides of the BRAM.
Data Width	Y	Port A side write and read port data width. We currently only support Port A data width being a ratio of 1:2n or 2n:1 with Port B data width. The max ratio is 1:32 or 32:1. This field imposes limitations on the Port B side data width and address depth.
Address Depth	Y	Port A side address depth is the total number of data words accessible via Port A. This field imposes limitations on the Port B side data width and address depth. The Port B data width must be a valid integer ratio of the Port A data width.
Write Mode	Y	The write mode can be set to No Change in order to keep the read port value constant until the next read. It can be set to Write First to allow the write data to be seen on the read port before the next read.
Clock Polarity	Y	The write port clock polarity can be set to use either rising edge assignment or falling edge assignment.
Port Enable Active-High	Y	When this is enabled, the port enable *pea* is active-high. Otherwise, the port enable will be active-low.
Output Latch Reset Active-High	Y	When this is enabled, the output latch has an active-high synchronous reset. Otherwise, the output latch reset will be active-low.
Output Register Enabled	Y	When the Output Register is enabled, there is an additional cycle of latency for each read operation.
Output Register		
Clock Enable Priority	Y	The Clock Enable Priority defines the priority of the *outregcea* clock enable input relative to the *rstrega* reset input during an assertion of the *rstrega* signal on the output register of Port A. The value **rstreg** allows the Port A output register to be set/reset at the next active edge of the Port A clock without requiring a specific value on the *outregcea* output register clock enable input. The value **regce** requires that the *outregcea* output register clock enable input is high for the output register set/reset operation to occur at the next active edge of the Port A clock.
Output Register Reset Active-High	Y	When this is enabled, the output register has an active-high synchronous reset. Otherwise, the output register reset will be active-low.

Option	Editable	Description
Port B		
Port B Configuration	Y	BRAMs can be configured for read, write, or read/write capability independently on both Port A and Port B sides of the BRAM.
Data Width	Y	Port A side write and read port data width. We currently only support Port A data width being a ratio of 1:2n or 2n:1 with Port B data width. The max ratio is 1:32 or 32:1. This field imposes limitations on the Port B side data width and address depth.
Address Depth		Port A side address depth is the total number of data words accessible via Port A. This field imposes limitations on the Port B side data width and address depth. The Port B data width must be a valid integer ratio of the Port A data width.
Write Mode	Y	The write mode can be set to No Change in order to keep the read port value constant until the next read. It can be set to Write First to allow the write data to be seen on the read port before the next read.
Clock Polarity	Y	The write port clock polarity can be set to use either rising edge assignment or falling edge assignment.
Port Enable Active-High	Y	When this is enabled, the port enable *peb* is active-high. Otherwise, the port enable will be active-low.
Output Latch Reset Active-High	Y	When this is enabled, the output latch has an active-high synchronous reset. Otherwise, the output latch reset will be active-low.
Output Register Enabled	Y	When the Output Register is enabled, there is an additional cycle of latency for each read operation.
Output Register		
Clock Enable Priority	Y	The Clock Enable Priority defines the priority of the *outregceb* clock enable input relative to the *rstregb* reset input during an assertion of the *rstregb* signal on the output register of Port B. The value **rstreg** allows the Port B output register to be set/reset at the next active edge of the Port B clock without requiring a specific value on the *outregceb* output register clock enable input. The value **regce** requires that the *outregceb* output register clock enable input is high for the output register set/reset operation to occur at the next active edge of the Port B clock.
Output Register Reset Active-High	Y	When this is enabled, the output register has an active-high synchronous reset. Otherwise, the output register reset will be active-low.
Total Memory Size		Port A Address Depth x Port A Data Width

Option	Editable	Description
Number of BRAMs Used		The total number of BRAM instances which will be used to create a BRAM wrapper of the configured width(s) and depth(s).

Speedster22i DDR3 Configuration Editor

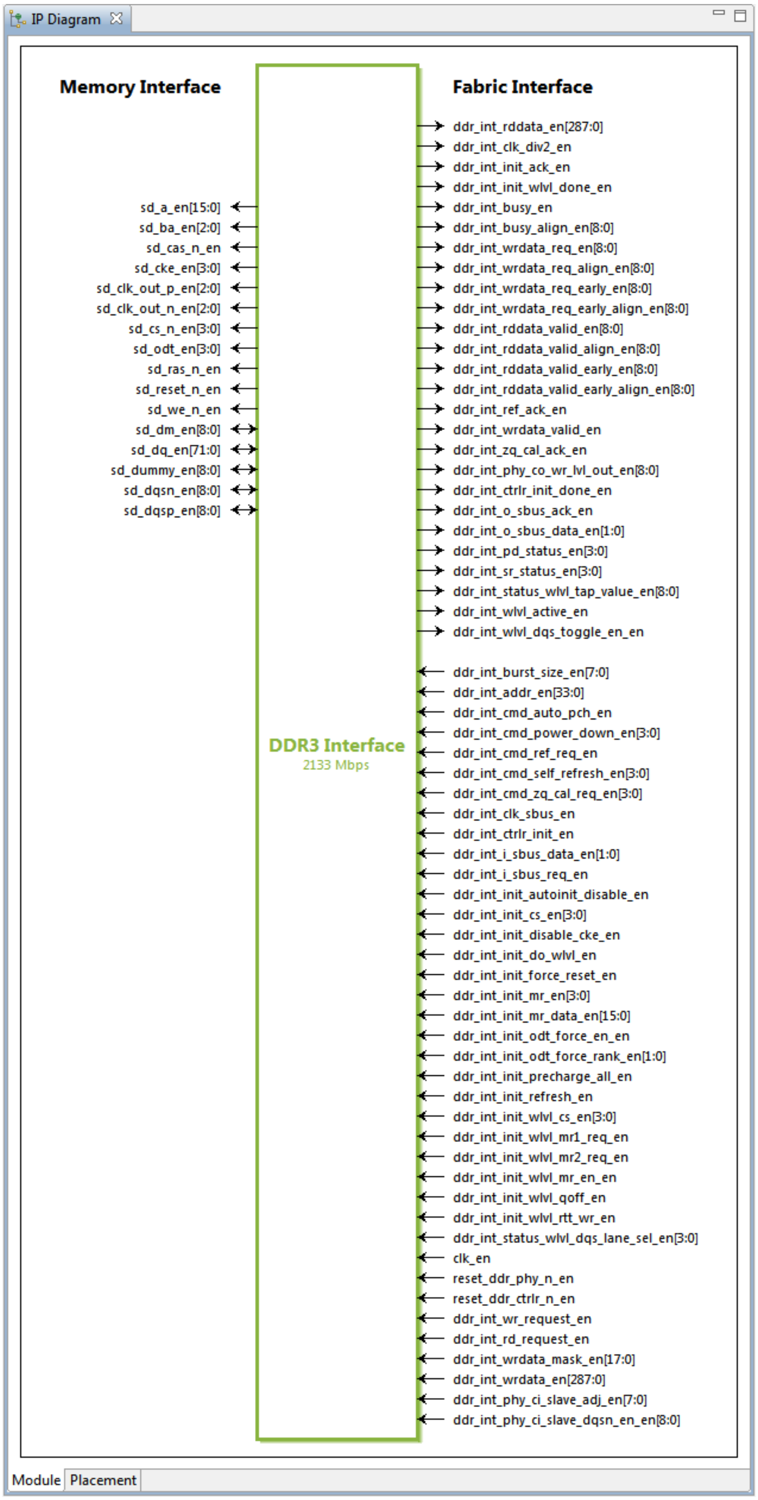
The DDR3 Configuration Editor provides a graphical wizard for creating a DDR3 Interface IP configuration file (`.acxip`). This editor allows the user to generate the required configuration files for designs requiring the embedded DDR3 controllers. See Creating an IP Configuration (see page 387).

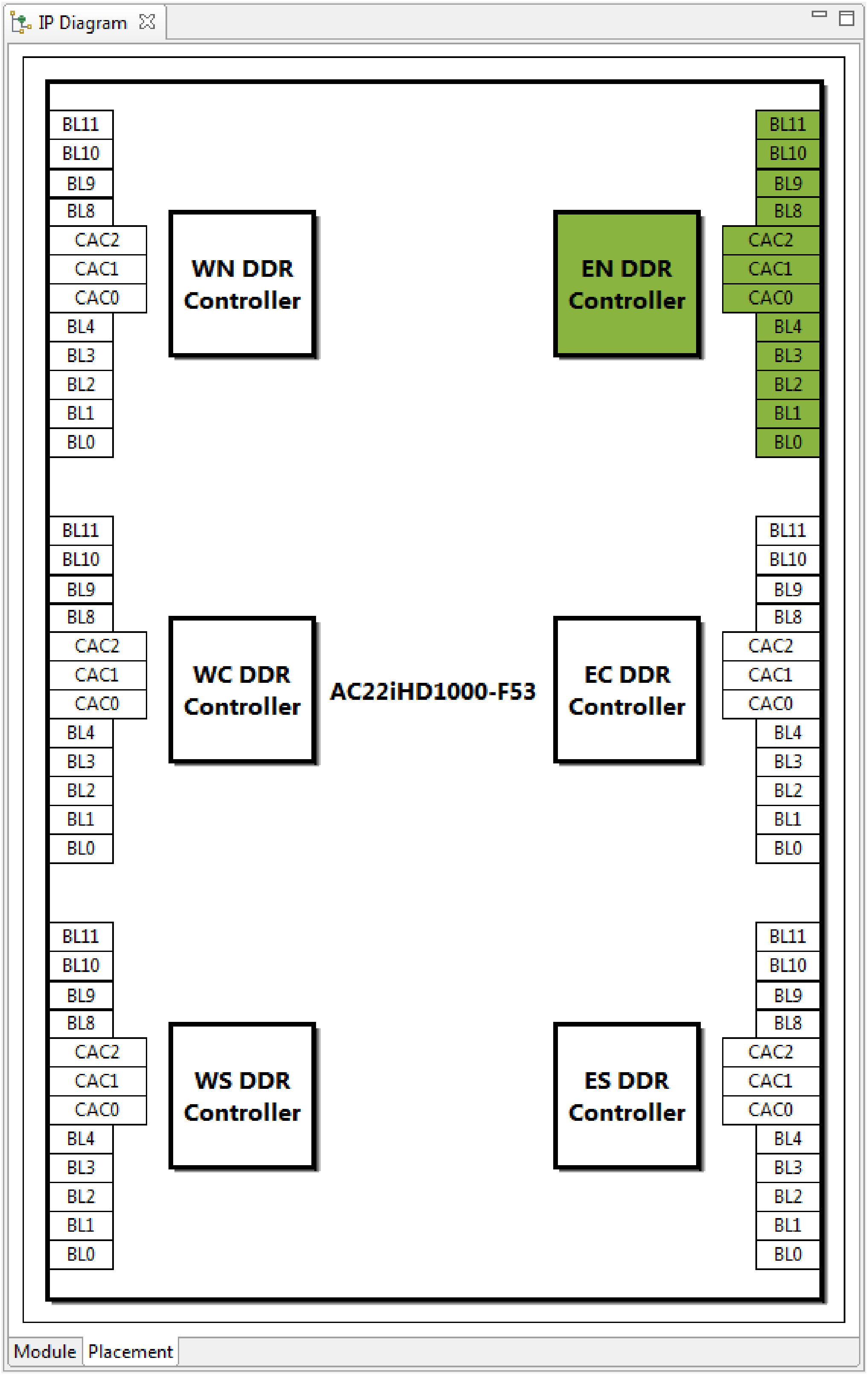
By default, the DDR3 Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**). The DDR3 Interface configuration information is broken up into several pages, organized by concept.

Once the user has configured the DDR Interface to meet their requirements, and the DDR3 Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

DDR3 IP Diagrams

Note that the DDR3 Configuration Editors support two subtabs in the IP Diagram View (see page 196): a **Module** tab showing the detailed DDR3 Interface based upon the current configuration, and a **Placement** tab showing the placement of the DDR3 Controller within the chosen **Target Device**, as selected on the Overview Page (see page 54). Both of these diagram tabs will update on-the-fly to match the current DDR3 configuration in the editor pages.


IP Diagram
Memory Interface
Fabric Interface
ddr_int_rddata_en[287:0]
ddr_int_clk_div2_en
ddr_int_init_ack_en
ddr_int_init_wlvl_done_en
ddr_int_busy_en
ddr_int_busy_align_en[8:0]
ddr_int_wrdata_req_en[8:0]
ddr_int_wrdata_req_align_en[8:0]
ddr_int_wrdata_req_early_en[8:0]
ddr_int_wrdata_req_early_align_en[8:0]
ddr_int_rddata_valid_en[8:0]
ddr_int_rddata_valid_align_en[8:0]
ddr_int_rddata_valid_early_en[8:0]
ddr_int_rddata_valid_early_align_en[8:0]
ddr_int_ref_ack_en
ddr_int_wrdata_valid_en
ddr_int_zq_cal_ack_en
ddr_int_phy_co_wr_lvl_out_en[8:0]
ddr_int_ctrlr_init_done_en
ddr_int_o_sbus_ack_en
ddr_int_o_sbus_data_en[1:0]
ddr_int_pd_status_en[3:0]
ddr_int_sr_status_en[3:0]
ddr_int_status_wlvl_tap_value_en[8:0]
ddr_int_wlvl_active_en
ddr_int_wlvl_dqs_toggle_en_en
sd_a_en[15:0]
sd_ba_en[2:0]
sd_cas_n_en
sd_cke_en[3:0]
sd_clk_out_p_en[2:0]
sd_clk_out_n_en[2:0]
sd_cs_n_en[3:0]
sd_odt_en[3:0]
sd_ras_n_en
sd_reset_n_en
sd_we_n_en
sd_dm_en[8:0]
sd_dq_en[71:0]
sd_dummy_en[8:0]
sd_dqsn_en[8:0]
sd_dqsp_en[8:0]
ddr_int_burst_size_en[7:0]
ddr_int_addr_en[33:0]
ddr_int_cmd_auto_pch_en
ddr_int_cmd_power_down_en[3:0]
ddr_int_cmd_ref_req_en
ddr_int_cmd_self_refresh_en[3:0]
ddr_int_cmd_zq_cal_req_en[3:0]
ddr_int_clk_sbus_en
ddr_int_ctrlr_init_en
ddr_int_i_sbus_data_en[1:0]
ddr_int_i_sbus_req_en
ddr_int_init_autoinit_disable_en
ddr_int_init_cs_en[3:0]
ddr_int_init_disable_cke_en
ddr_int_init_do_wlvl_en
ddr_int_init_force_reset_en
ddr_int_init_mr_en[3:0]
ddr_int_init_mr_data_en[15:0]
ddr_int_init_odt_force_en_en
ddr_int_init_odt_force_rank_en[1:0]
ddr_int_init_precharge_all_en
ddr_int_init_refresh_en
ddr_int_init_wlvl_cs_en[3:0]
ddr_int_init_wlvl_mr1_req_en
ddr_int_init_wlvl_mr2_req_en
ddr_int_init_wlvl_mr_en_en
ddr_int_init_wlvl_qoff_en
ddr_int_init_wlvl_rtt_wr_en
ddr_int_status_wlvl_dqs_lane_sel_en[3:0]
clk_en
reset_ddr_phy_n_en
reset_ddr_ctrlr_n_en
ddr_int_wr_request_en
ddr_int_rd_request_en
ddr_int_wrdata_mask_en[17:0]
ddr_int_wrdata_en[287:0]
ddr_int_phy_ci_slave_adj_en[7:0]
ddr_int_phy_ci_slave_dqsn_en_en[8:0]
DDR3 Interface
2133 Mbps
Module
Placement


IP Diagram
WN DDR Controller
EN DDR Controller
WC DDR Controller
AC22iHD1000-F53
EC DDR Controller
WS DDR Controller
ES DDR Controller
BL11
BL10
BL9
BL8
CAC2
CAC1
CAC0
BL4
BL3
BL2
BL1
BL0
Module
Placement

DDR3 Overview Page

The Overview page of the Speedster22i DDR3 Configuration Editor (see page 51) contains the top-level, global properties that govern the structure and base configuration of the DDR3 Interface.

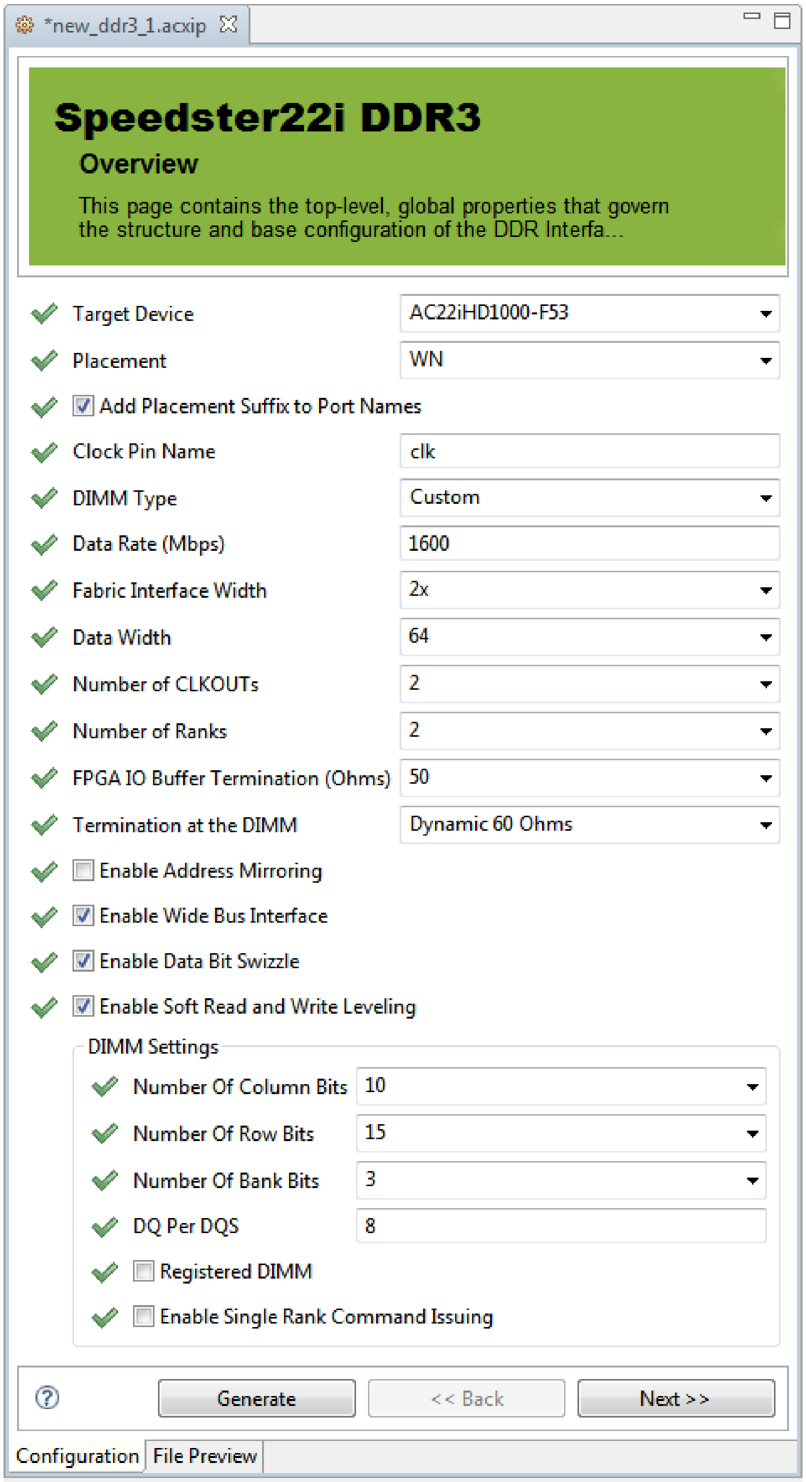


Table 14: *DDR3 Editor Overview Page Options*

Option	Description
Target Device	Allows the user to select from the Achronix devices that support this IP.
Placement	Select the location on the chip where this DDR interface should be placed. (Alternately, left-click on the desired Controller in the Placement Diagram of the IP Diagram View (see page 196).)
Add Placement Suffix to Port Names	Enabling this option causes the generated RTL wrapper and constraints to attach a suffix to all port names based on placement. This helps uniquify the top-level IOs in the design. For example, if you place your DDR interface on the East North site, `"_en"` would get added to the end of all the port names.
Clock Pin Name	Enter the reference clock pin name. Will be used to generate clock constraints. May be a top level design pin, a PLL clock output pin, etc.
DIMM Type	Select from a predefined library of standard DIMMs, or select Custom for full customization options.
Data Rate (Mbps)	May be configured to use a standard rate, or may be customized. The max data rate for 1x Fabric Interface Width is 1066Mbps. The max data rate for a Fabric Interface Width of 2x is 1600Mbps.
Fabric Interface Width	The fabric interface width can be set to 2x to enable a wider Core interface that runs at half speed. This is needed for higher data rates, such as 1600 Mbps.
Data Width	Local side data width.
Number of CLKOUTs	Number of DDR3 DIMM clocks.
Number of Ranks	Number of chip selects used.
FPGA IO Buffer Termination (Ohms)	The FPGA IO buffer termination is specified to ensure that the trace impedance is matched and that reflections are minimized during read operations.
Termination at the DIMM	Termination settings at the DIMM configured through MR registers. The options are to enable or disable dynamic ODT, and the termination value to be used when performing write operations.
Enable Address Mirroring	Enable Address Mirroring in the DDR Controller.
Enable Wide Bus Interface	When enabled, this doubles the width of the data bus, and helps meet timing at higher frequencies.

Option		Description
Enable Data Bit Swizzle		Swizzle enables data bits from the fabric to the IO ring to be re-arranged to optimize for routing. This is especially important at high data rates, as it will help significantly with timing closure. Please note that when swizzle is enabled, data masking will not be available.
Enable Soft Read and Write Leveling		Enabling soft read and write leveling ensures that the algorithms to perform the read and write leveling functions are built in the programmable logic fabric and seamlessly integrated into the ddr3 macro. This option allows the PHY to take advantage of a larger delay range, and prevents the need for manual tuning of the PHY clk/cac delay values. This option prevents the need for manual tuning of the read delay parameters for every bit in the data byte lanes.
DIMM Settings		
	Number of Column Bits	Number of bits for Column Address.
	Number of Row Bits	Number of bits for Row Address.
	Number of Bank Bits	Number of bits for Bank Address.
	DQ Per DQS	Number of DQ bits per DQS line.
	Registered DIMM	Whether or not the DDR3 DIMM is registered.
	Enable Single Rank Command Issuing	When set, controller only issues commands to one rank at a time (i.e. only one chip select is ever asserted). This needs to be set for RDIMMs, and can optionally be set for UDIMMs for staggered refresh commands to reduce instantaneous current.

Memory Timing Page

The Memory Timing Page allows the user to configure the memory timings for the DDR3 Interface.

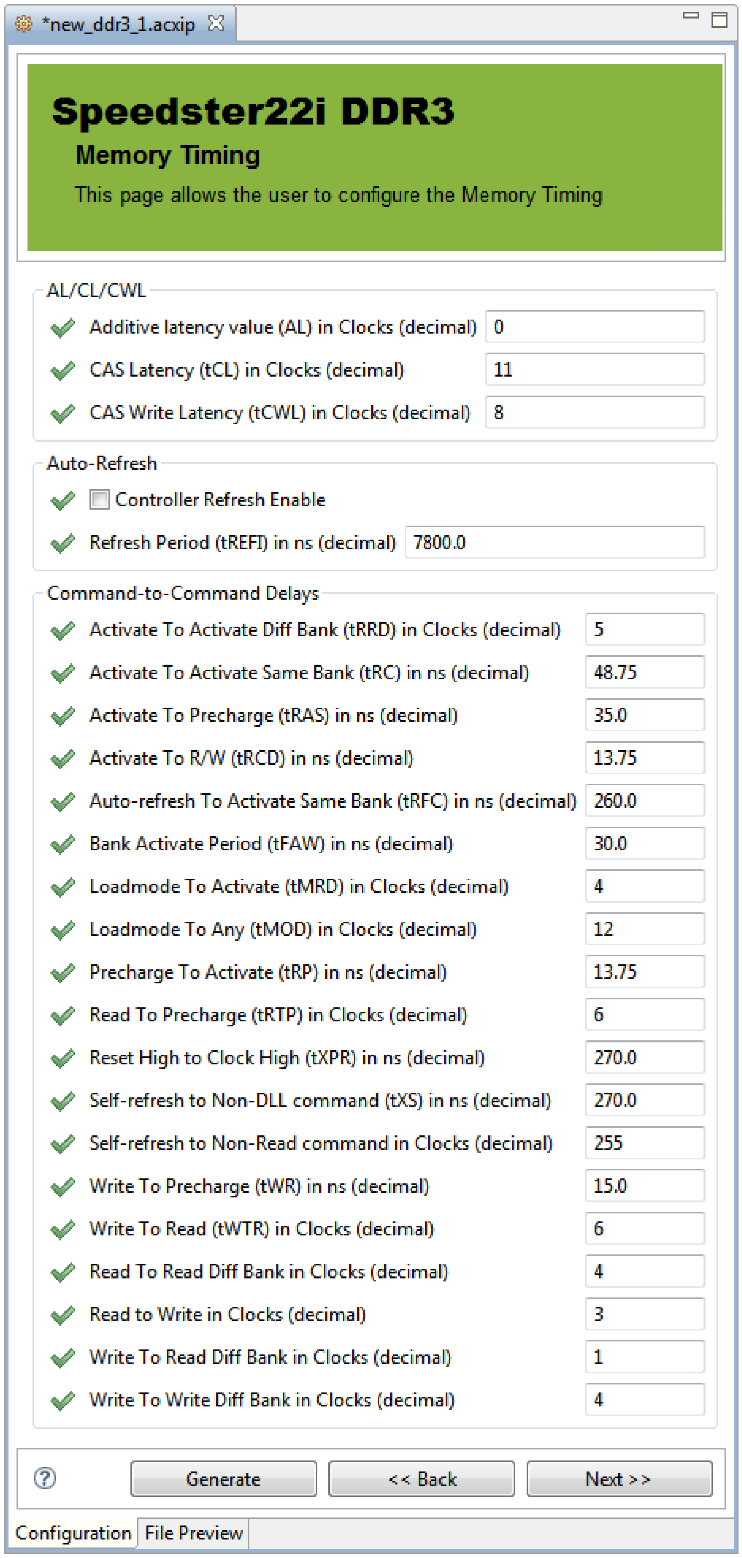

*new_ddr3_1.acxip
Speedster22i DDR3
Memory Timing
This page allows the user to configure the Memory Timing
AL/CL/CWL
Additive latency value (AL) in Clocks (decimal) 0
CAS Latency (tCL) in Clocks (decimal) 11
CAS Write Latency (tCWL) in Clocks (decimal) 8
Auto-Refresh
Controller Refresh Enable
Refresh Period (tREFI) in ns (decimal) 7800.0
Command-to-Command Delays
Activate To Activate Diff Bank (tRRD) in Clocks (decimal) 5
Activate To Activate Same Bank (tRC) in ns (decimal) 48.75
Activate To Precharge (tRAS) in ns (decimal) 35.0
Activate To R/W (tRCD) in ns (decimal) 13.75
Auto-refresh To Activate Same Bank (tRFC) in ns (decimal) 260.0
Bank Activate Period (tFAW) in ns (decimal) 30.0
Loadmode To Activate (tMRD) in Clocks (decimal) 4
Loadmode To Any (tMOD) in Clocks (decimal) 12
Precharge To Activate (tRP) in ns (decimal) 13.75
Read To Precharge (tRTP) in Clocks (decimal) 6
Reset High to Clock High (tXPR) in ns (decimal) 270.0
Self-refresh to Non-DLL command (tXS) in ns (decimal) 270.0
Self-refresh to Non-Read command in Clocks (decimal) 255
Write To Precharge (tWR) in ns (decimal) 15.0
Write To Read (tWTR) in Clocks (decimal) 6
Read To Read Diff Bank in Clocks (decimal) 4
Read to Write in Clocks (decimal) 3
Write To Read Diff Bank in Clocks (decimal) 1
Write To Write Diff Bank in Clocks (decimal) 4
Generate
<< Back
Next >>
Configuration
File Preview

Table 15: *DDR3 Editor Memory Timing Page Options*

Option	Description
AL/CL/CWL	
Additive latency value (AL) in Clocks (decimal)	AL in clock cycles
CAS Latency (tCL) in Clocks (decimal)	CAS Latency (tCL) in clock cycles
CAS Write Latency (tCWL) in Clocks (decimal)	CAS Write Latency (tCWL) in clock cycles
Auto-Refresh	
Controller Refresh Enable	When enabled, the DDR Controller will handle DDR Memory Refreshes
Refresh Period (tREFI) in ns (decimal)	Duration between Refresh commands (tREFI) in nanoseconds
Command-to-Command Delays	
Activate To Activate Diff Bank (tRRD) in Clocks (decimal)	Minimum clock cycles between ACTIVATE to ACTIVATE in different banks (tRRD)
Activate To Activate Same Bank (tRC) in ns (decimal)	Minimum duration from Activate to Activate/Auto-refresh in same bank (tRC) in nanoseconds
Activate To Precharge (tRAS) in ns (decimal)	Minimum duration from Activate to Precharge (tRAS) in nanoseconds
Activate To R/W (tRCD) in ns (decimal)	Minimum duration between Activate and Read/Write (tRCD) in nanoseconds
Auto-refresh To Activate Same Bank (tRFC) in ns (decimal)	Minimum duration in nanoseconds from Auto Refresh to Activate/Auto Refresh in the same Bank
Bank Activate Period (tFAW) in ns (decimal)	Four bank activate period (tFAW) in nanoseconds
Loadmode To Activate (tMRD) in Clocks (decimal)	Minimum clock cycles from Loadmode to Activate command (tMRD)
Loadmode To Any (tMOD) in Clocks (decimal)	Minimum clock cycles from Loadmode to Any command (tMOD)
Precharge To Activate (tRP) in ns (decimal)	Minimum duration between Precharge to Activate in nanoseconds
Read To Precharge (tRTP) in Clocks (decimal)	Minimum clock cycles from Read to Precharge (tRTP)

	Option	Description
	Reset High to Clock High (tXPR) in ns (decimal)	Minimum duration from memory reset high to cke high (tXPR) in nanoseconds
	Self-refresh to Non-DLL command (tXS) in ns (decimal)	Minimum duration from Self-refresh to Non-DLL command in nanoseconds
	Self-refresh to Non-Read command in Clocks (decimal)	Minimum clock cycles from Self-refresh to Non-Read command
	Write To Precharge (tWR) in ns (decimal)	Minimum duration from write to Precharge (tWR) in nanoseconds
	Write To Read (tWTR) in Clocks (decimal)	Minimum clock cycles from Write to Read (tWTR)
	Read To Read Diff Bank in Clocks (decimal)	Minimum clock cycles from Read to Read (different banks)
	Read to Write in Clocks (decimal)	Read to Write delay in clock cycles
	Write To Read Diff Bank in Clocks (decimal)	Minimum clock cycles from Write to Read (different banks)
	Write To Write Diff Bank in Clocks (decimal)	Minimum clock cycles from Write to Write (different banks)

DLL Timing Page

The DLL Timing Page of the Speedster22i DDR3 Configuration Editor (see page 51) allows the user to configure the DLL timing parameters for the DDR3 Interface.

The number of data entry fields on this page will vary according to the number of Byte Lanes used by the memory interface. Each used byte lane will have a DQ Slave Adjust value for each of the 8 bits, along with a single DQS Slave Adjust value.

For simplicity, the following screenshot will show a single Byte Lane configuration, which is only valid for a (custom) data width of 8.

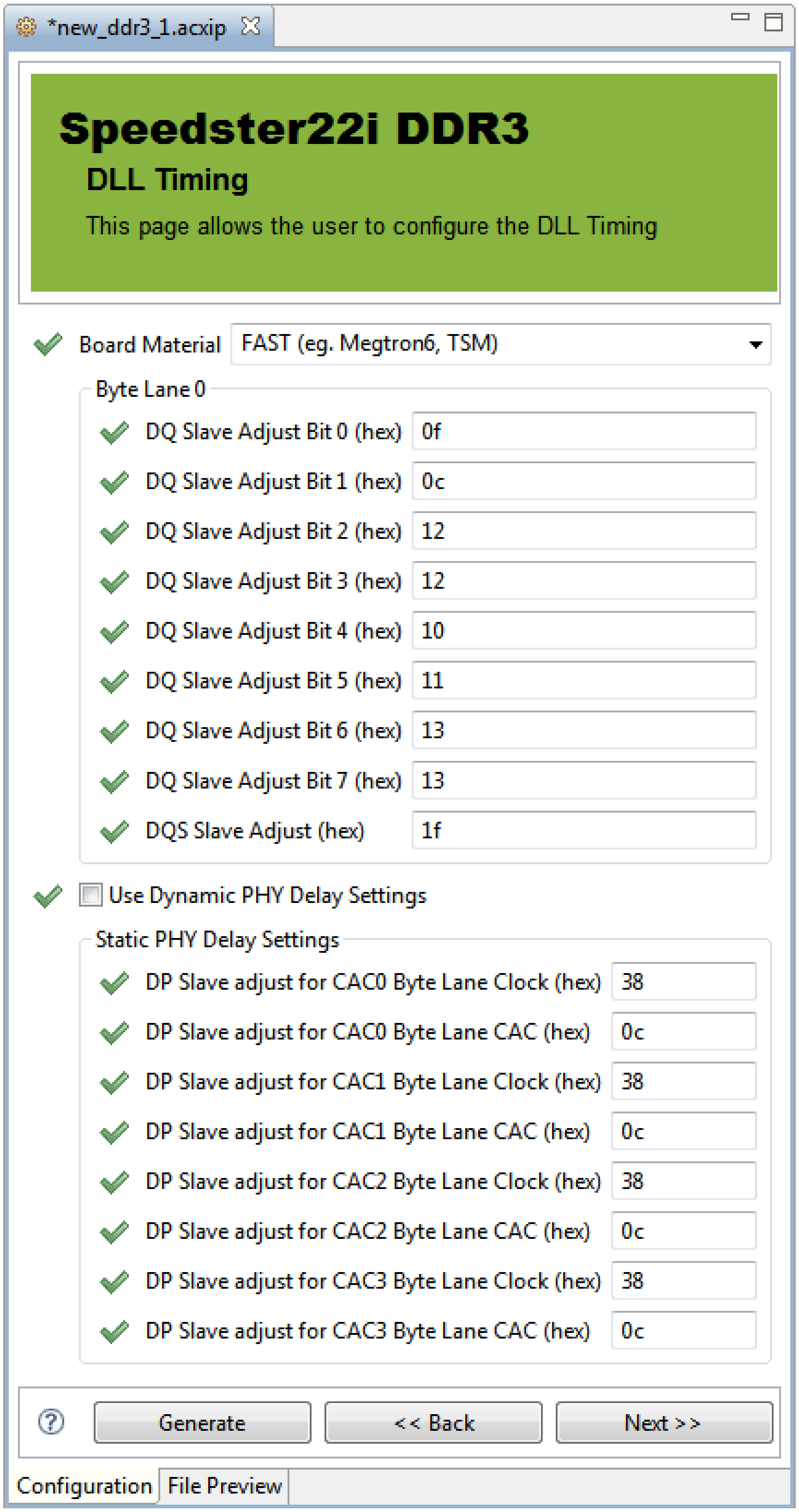
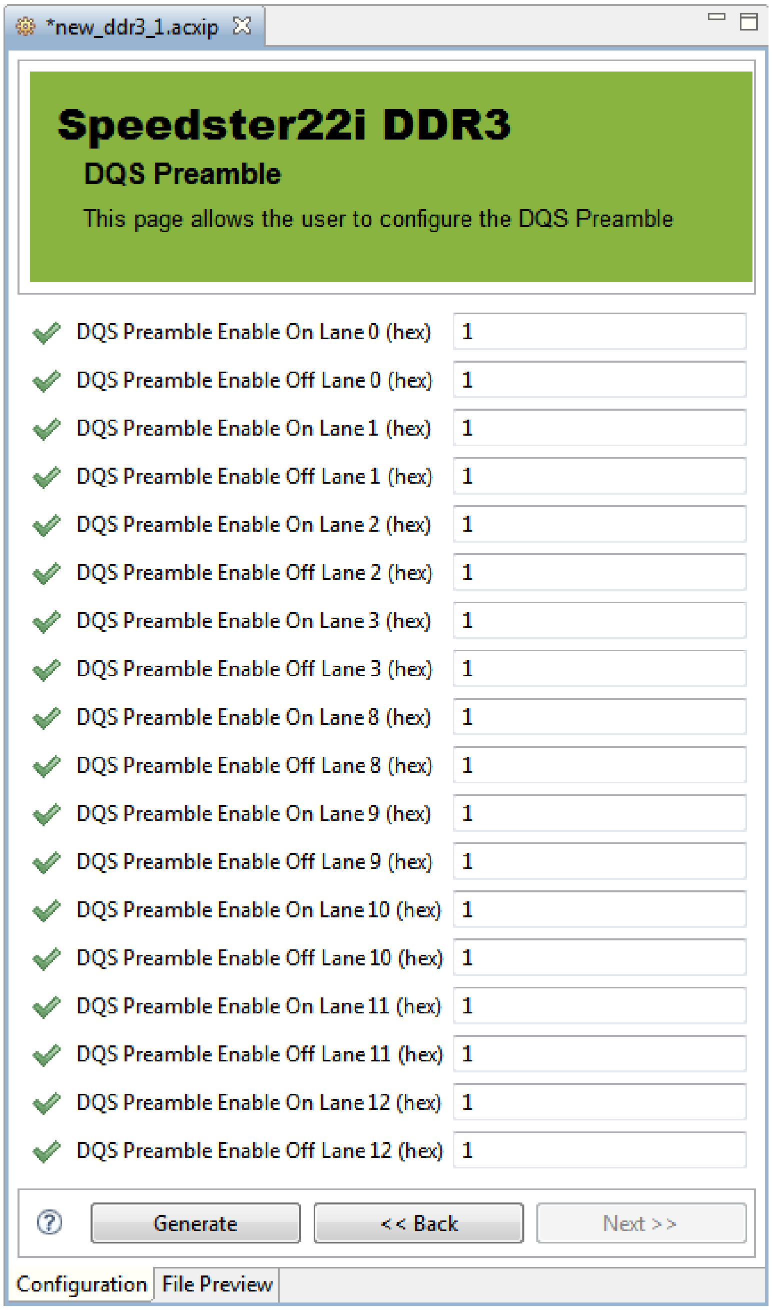

*new_ddr3_1.acxip
Speedster22i DDR3
DLL Timing
This page allows the user to configure the DLL Timing
Board Material
FAST (eg. Megtron6, TSM)
Byte Lane 0
DQ Slave Adjust Bit 0 (hex)
0f
DQ Slave Adjust Bit 1 (hex)
0c
DQ Slave Adjust Bit 2 (hex)
12
DQ Slave Adjust Bit 3 (hex)
12
DQ Slave Adjust Bit 4 (hex)
10
DQ Slave Adjust Bit 5 (hex)
11
DQ Slave Adjust Bit 6 (hex)
13
DQ Slave Adjust Bit 7 (hex)
13
DQS Slave Adjust (hex)
1f
Use Dynamic PHY Delay Settings
Static PHY Delay Settings
DP Slave adjust for CAC0 Byte Lane Clock (hex)
38
DP Slave adjust for CAC0 Byte Lane CAC (hex)
0c
DP Slave adjust for CAC1 Byte Lane Clock (hex)
38
DP Slave adjust for CAC1 Byte Lane CAC (hex)
0c
DP Slave adjust for CAC2 Byte Lane Clock (hex)
38
DP Slave adjust for CAC2 Byte Lane CAC (hex)
0c
DP Slave adjust for CAC3 Byte Lane Clock (hex)
38
DP Slave adjust for CAC3 Byte Lane CAC (hex)
0c
Generate
<< Back
Next >>
Configuration
File Preview

Table 16: *DDR3 Editor DLL Timing Page Options*

Option		Description
Board Material		Board material and associated properties are used for delay calculations.
Byte Lane *N*		
	DQ Slave Adjust Bit 0 (hex)	DLL delay adjust value for DQ bit 0
	...	...
	DQ Slave Adjust Bit 7 (hex)	DLL delay adjust value for DQ bit 7
	DQS Slave Adjust (hex)	DLL delay adjust value for all DQS lines in byte_lane*N*
Use Dynamic PHY Delay Settings		Static PHY delay setting enables a bitstream specific set of Clk and Cac delay values to be provided to the controller as specified below. These are used to ensure that write-leveling values for each bytelane fall within a desired range. Dynamic PHY delay setting provides user the ability to change these in user mode by feeding the appropriate values through a fabric interface.
Static PHY Delay Settings		
	DP Slave adjust for CAC0 Byte Lane Clock (hex)	DP Slave adjust for CAC Byte Lane Clock
	DP Slave adjust for CAC0 Byte Lane CAC (hex)	DP Slave adjust for CAC Byte Lane CAC
	...	...
	DP Slave adjust for CAC3 Byte Lane Clock (hex)	DP Slave adjust for CAC Byte Lane Clock
	DP Slave adjust for CAC3 Byte Lane CAC (hex)	DP Slave adjust for CAC Byte Lane CAC

DQS Preamble Page

The DQS Preamble page of the Speedster22i DDR3 Configuration Editor (see page 51) allows the user to configure the DQS Preamble settings of the interface.



Option	Description
DQS Preamble Enable On Lane *N* (hex)	DQS Preamble Enable On Lane *N*
DQS Preamble Enable Off Lane *N* (hex)	DQS Preamble Enable Off Lane *N*

Speedster22i Ethernet Configuration Editor

The Ethernet interface configuration editor provides a simple graphical editor used to configure the SerDes interface for the Ethernet protocol, and saves the user configuration in an Ethernet IP configuration file (.acxip). See Creating an IP Configuration (see page 387).

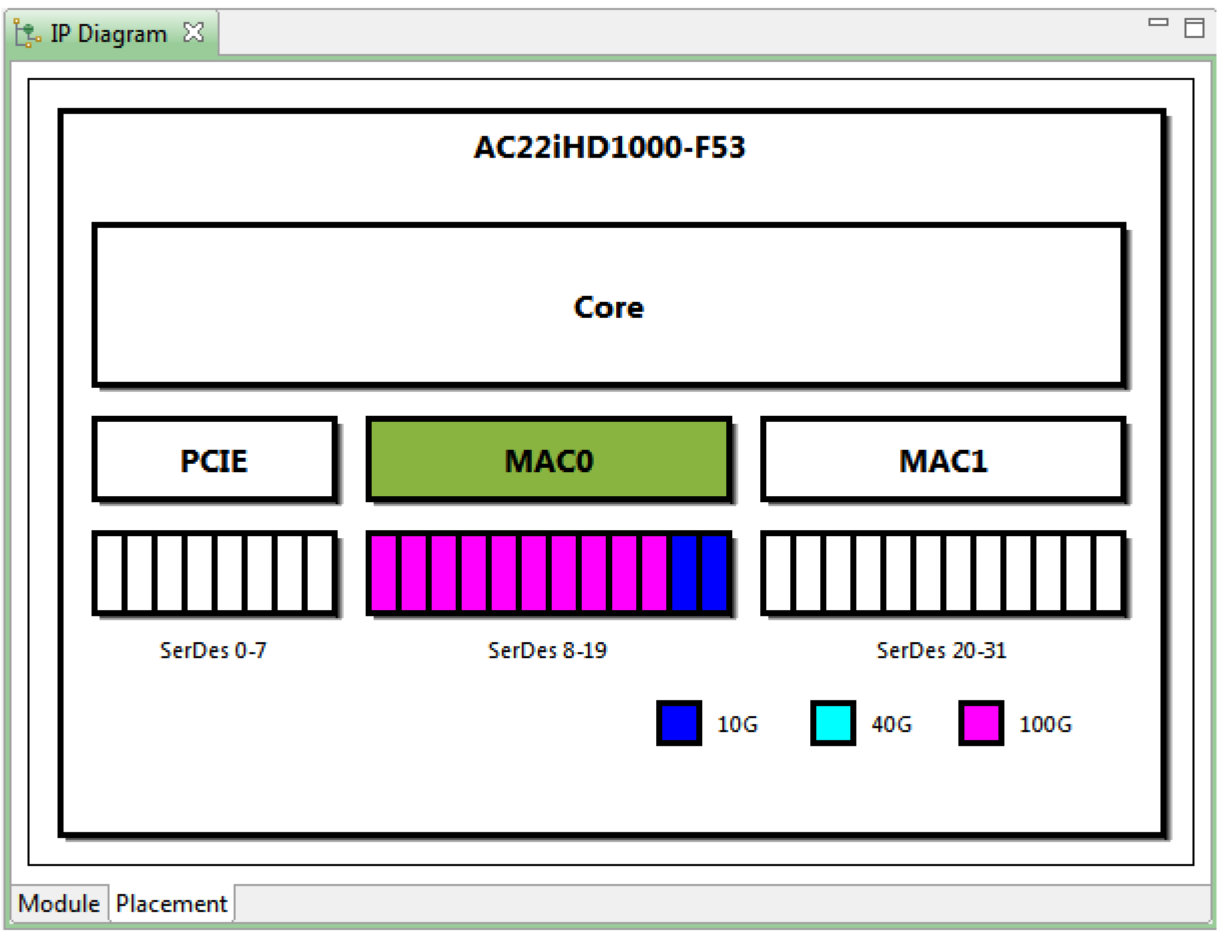
By default, the Ethernet Configuration Editor is included in the IP Configuration perspective (**Window → Open Perspective → IP Configuration**).

Once the user has configured the Ethernet interface to meet their requirements, and the Ethernet Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

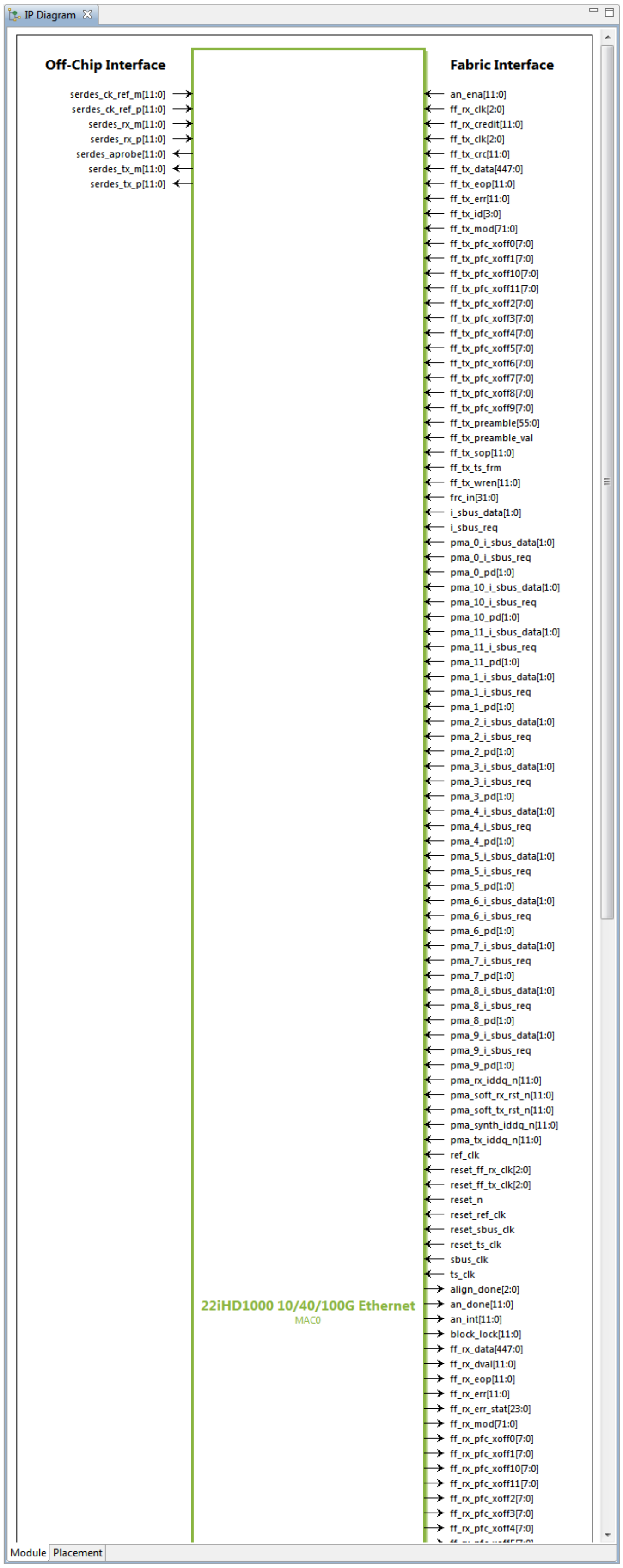
IP Diagrams

Note that the Ethernet Configuration Editors support two sub-tabs in the IP Diagram View (see page 196): a Module tab showing the detailed DDR3 Interface based upon the current configuration, and a Placement tab showing the placement of the DDR3 Controller within the chosen "Target Device", as selected on the Overview Page (see page 65). Both of these diagram tabs will update on-the-fly to match the current DDR3 configuration in the editor pages.

The example Placement Diagram shows an AC22iHD1000 device in the F53 package, with the Ethernet instance placed in MAC0, containing a single 100G interface and two 10G interfaces.



The Module diagram shows all the inputs and outputs of the module instantiation as configured. The actual inputs and outputs listed can change according to the needs of the chosen configuration.


IP Diagram
Off-Chip Interface
serdes_ck_ref_m[11:0]
serdes_ck_ref_p[11:0]
serdes_rx_m[11:0]
serdes_rx_p[11:0]
serdes_aprobe[11:0]
serdes_tx_m[11:0]
serdes_tx_p[11:0]
Fabric Interface
an_ena[11:0]
ff_rx_clk[2:0]
ff_rx_credit[11:0]
ff_tx_clk[2:0]
ff_tx_crc[11:0]
ff_tx_data[447:0]
ff_tx_eop[11:0]
ff_tx_err[11:0]
ff_tx_id[3:0]
ff_tx_mod[71:0]
ff_tx_pfc_xoff0[7:0]
ff_tx_pfc_xoff1[7:0]
ff_tx_pfc_xoff10[7:0]
ff_tx_pfc_xoff11[7:0]
ff_tx_pfc_xoff2[7:0]
ff_tx_pfc_xoff3[7:0]
ff_tx_pfc_xoff4[7:0]
ff_tx_pfc_xoff5[7:0]
ff_tx_pfc_xoff6[7:0]
ff_tx_pfc_xoff7[7:0]
ff_tx_pfc_xoff8[7:0]
ff_tx_pfc_xoff9[7:0]
ff_tx_preamble[55:0]
ff_tx_preamble_val
ff_tx_sop[11:0]
ff_tx_ts_frm
ff_tx_wren[11:0]
frc_in[31:0]
i_sbus_data[1:0]
i_sbus_req
pma_0_i_sbus_data[1:0]
pma_0_i_sbus_req
pma_0_pd[1:0]
pma_10_i_sbus_data[1:0]
pma_10_i_sbus_req
pma_10_pd[1:0]
pma_11_i_sbus_data[1:0]
pma_11_i_sbus_req
pma_11_pd[1:0]
pma_1_i_sbus_data[1:0]
pma_1_i_sbus_req
pma_1_pd[1:0]
pma_2_i_sbus_data[1:0]
pma_2_i_sbus_req
pma_2_pd[1:0]
pma_3_i_sbus_data[1:0]
pma_3_i_sbus_req
pma_3_pd[1:0]
pma_4_i_sbus_data[1:0]
pma_4_i_sbus_req
pma_4_pd[1:0]
pma_5_i_sbus_data[1:0]
pma_5_i_sbus_req
pma_5_pd[1:0]
pma_6_i_sbus_data[1:0]
pma_6_i_sbus_req
pma_6_pd[1:0]
pma_7_i_sbus_data[1:0]
pma_7_i_sbus_req
pma_7_pd[1:0]
pma_8_i_sbus_data[1:0]
pma_8_i_sbus_req
pma_8_pd[1:0]
pma_9_i_sbus_data[1:0]
pma_9_i_sbus_req
pma_9_pd[1:0]
pma_rx_iddq_n[11:0]
pma_soft_rx_rst_n[11:0]
pma_soft_tx_rst_n[11:0]
pma_synth_iddq_n[11:0]
pma_tx_iddq_n[11:0]
ref_clk
reset_ff_rx_clk[2:0]
reset_ff_tx_clk[2:0]
reset_n
reset_ref_clk
reset_sbus_clk
reset_ts_clk
sbus_clk
ts_clk
align_done[2:0]
an_done[11:0]
an_int[11:0]
block_lock[11:0]
ff_rx_data[447:0]
ff_rx_dval[11:0]
ff_rx_eop[11:0]
ff_rx_err[11:0]
ff_rx_err_stat[23:0]
ff_rx_mod[71:0]
ff_rx_pfc_xoff0[7:0]
ff_rx_pfc_xoff1[7:0]
ff_rx_pfc_xoff10[7:0]
ff_rx_pfc_xoff11[7:0]
ff_rx_pfc_xoff2[7:0]
ff_rx_pfc_xoff3[7:0]
ff_rx_pfc_xoff4[7:0]
22iHD1000 10/40/100G Ethernet
MAC0
Module
Placement

Ethernet Overview Page

The Overview page of the Ethernet IP Configuration Editor (see page 63) contains the common options that govern the structure and configuration of the Ethernet interface. Options that may be lane-specific are on subsequent pages.

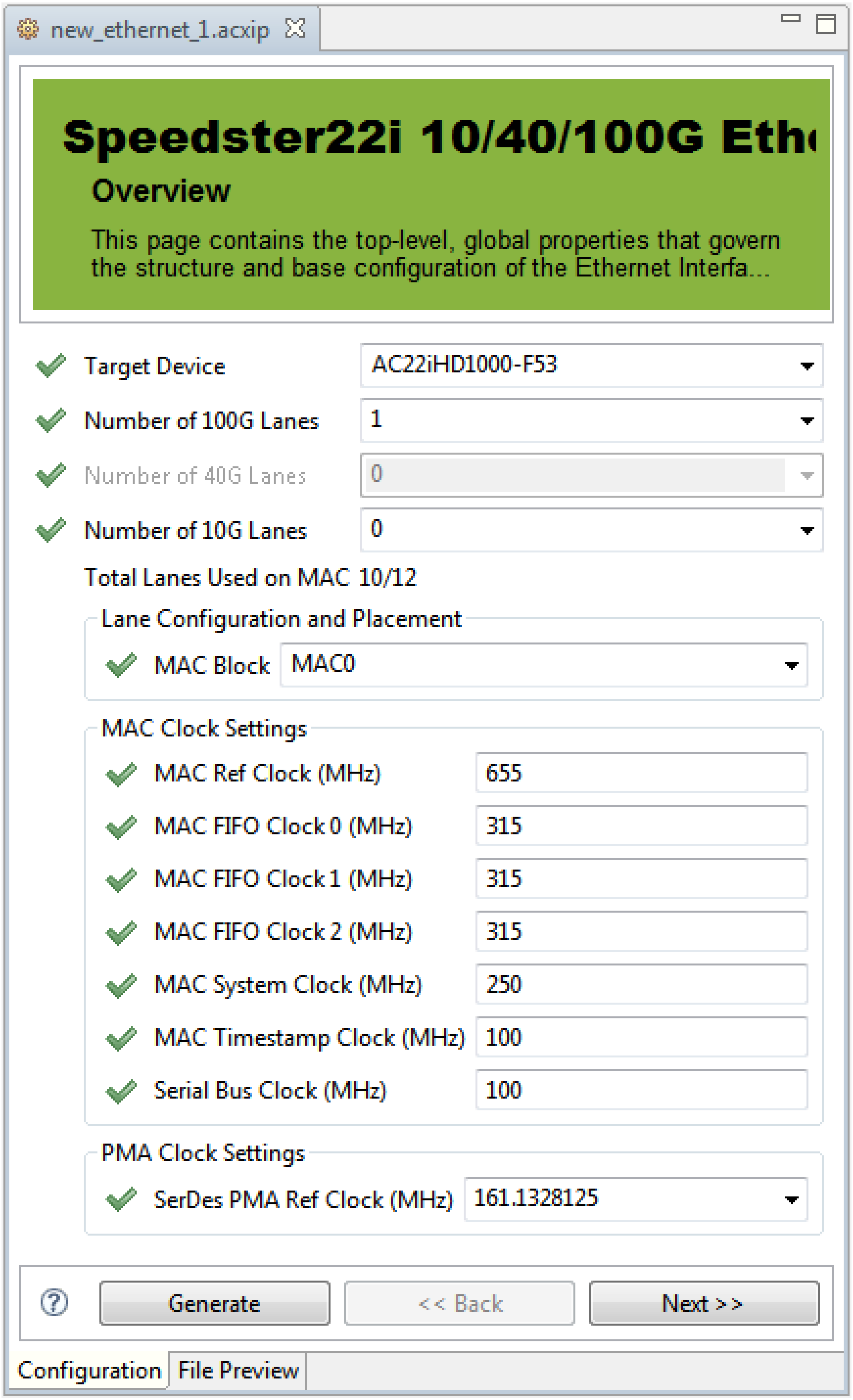


Table 17: *Ethernet Editor Overview Page Options*

Option		Description
Target Device		Select the intended 22i Target Device. This may affect choices in other option fields, and will affect the generated wrapper files.
Number of 100G Lanes		Each 100G lane consumes 10 10G lanes out of 12 available on the MAC; allowed range is 0 - 1.
Number of 40G Lanes		Each 40G lane consumes 4 10G lanes out of 12 available on the MAC; allowed range is 0 - 3.
Number of 10G Lanes		Each 10G lane consumes 1 10G lanes out of 12 available on the MAC; allowed range is 0 - 12.
Lane Configuration and Placement		
	MAC Block	There are two Ethernet MAC sites on the south side of the chip. See the Placement tab of the IP Diagram view for a graphical representation of the sites, including the automatic 100G / 40G / 10G lane assignments within the MAC. (Specific lane assignment within the MAC is currently not user-configurable.)
MAC Clock Settings		
	MAC Ref Clock (MHz)	The reference clock frequency of this Ethernet MAC (used to generate SDC).
	MAC FIFO Clock0 (MHz)	The MAC FIFO clock frequency (used to generate SDC); allowed range is 325.0 to 500.0.
	MAC FIFO Clock1 (MHz)	The MAC FIFO clock frequency (used to generate SDC); allowed range is 325.0 to 500.0.
	MAC FIFO Clock2 (MHz)	The MAC FIFO clock frequency (used to generate SDC); allowed range is 325.0 to 500.0.
	MAC System Clock (MHz)	The MAC system clock frequency (used to generate SDC).
	MAC Timestamp Clock (MHz)	The MAC timestamp clock frequency (used to generate SDC).

Option		Description
	Serial Bus Clock (MHz)	The Serial Bus clock frequency of this Ethernet MAC (used to generate SDC).
PMA Clock Settings		
	SerDes PMA Ref Clock (MHz)	The SerDes PMA reference clock frequency (used to generate SDC).

Ethernet PMA Settings Page

The PMA Settings page of the Speedster22i Ethernet Configuration Editor (see page 63) allows the user to choose whether the instantiated macro will include lane-specific settings for the Rx and Tx PMA, or shared settings across all lanes. The Rx and Tx PMA settings are configured separately.

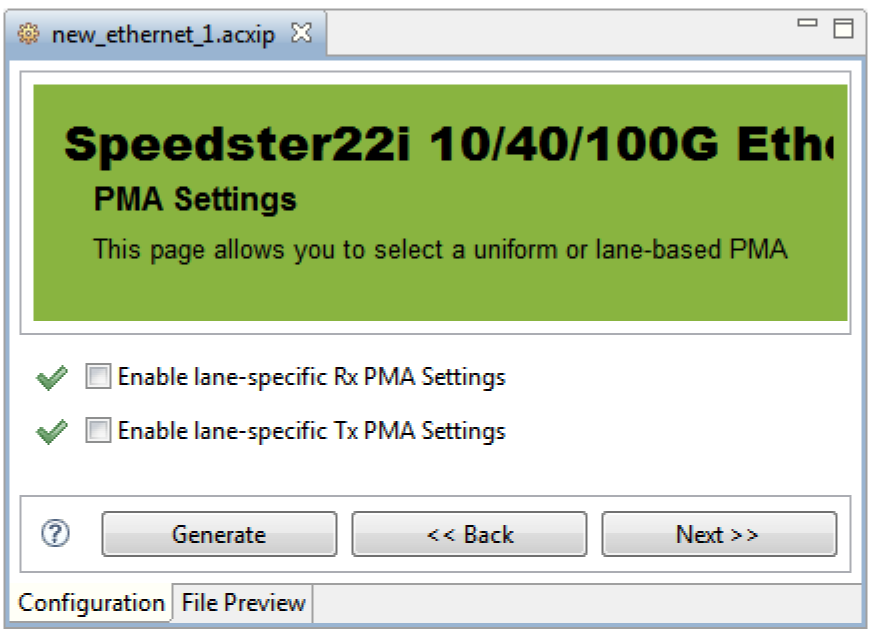


Table 18: *Ethernet Editor PMA Settings Page Options*

Option	Description
Enable lane-specific Rx PMA Settings	If selected, each active lane will have its own page of (potentially) unique Rx PMA Settings. If deselected, all active lanes will share the same Rx PMA Settings, presented on a single page.
Enable lane-specific Tx PMA Settings	If selected, each active lane will have its own page of (potentially) unique Tx PMA Settings. If deselected, all active lanes will share the same Tx PMA Settings, presented on a single page.

Ethernet Rx PMA Equalization Page

The Rx PMA Equalization Page(s) of the Speedster22i Ethernet Configuration Editor (see page 63) allow the user to set the per-lane or shared options for Rx PMA Equalization. (If per-lane settings are required, see the Ethernet PMA Settings Page (see page 67).)

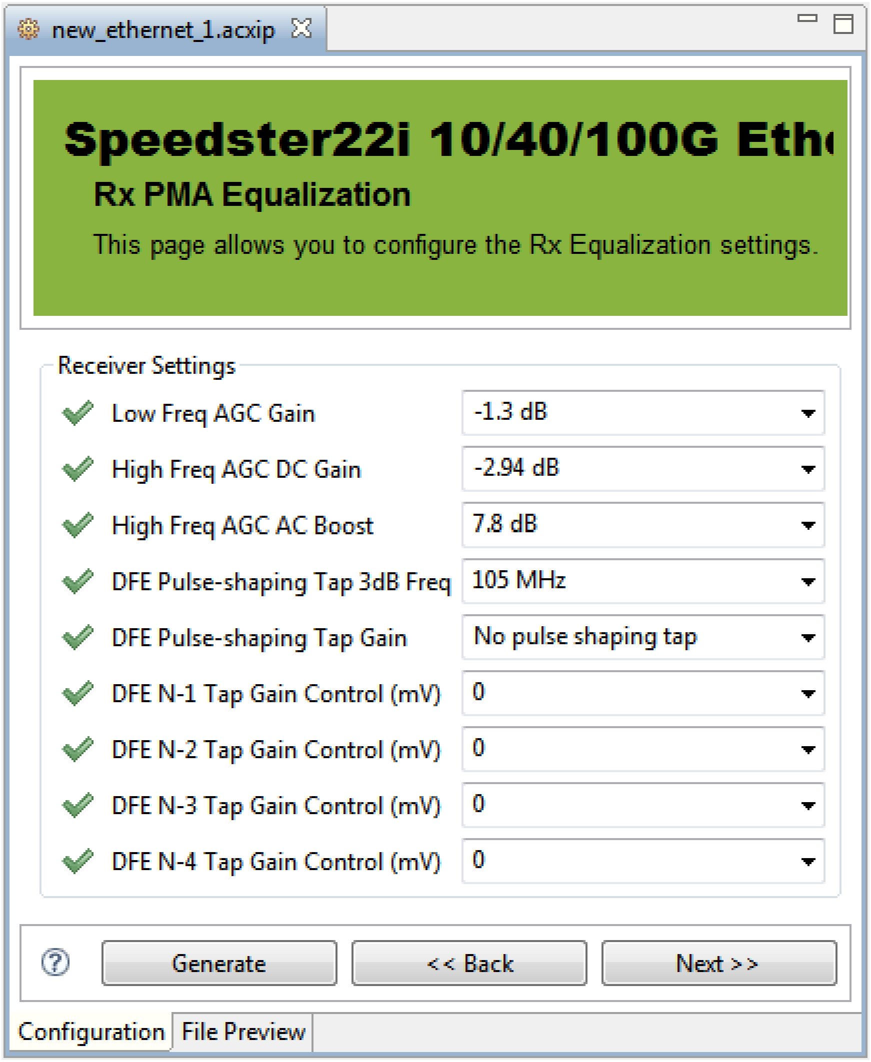


Table 19: *Ethernet Rx PMA Equalization Options*

Option	Description
Low Freq AGC Gain	Low frequency AGC Gain
High Freq AGC DC Gain	Rx AGC high frequency DC gain
High Freq AGC AC Boost	Rx AGC high frequency AC boost
DFE Pulse-shaping Tap 3dB Freq	DFE pulse-shaping tap 3dB frequency

Option	Description
DFE Pulse-shaping Tap Gain	DFE pulse-shaping tap gain
DFE N-1 Tap Gain Control (mV)	DFE tap 1 gain control
DFE N-2 Tap Gain Control (mV)	DFE tap 2 gain control
DFE N-3 Tap Gain Control (mV)	DFE tap 3 gain control
DFE N-4 Tap Gain Control (mV)	DFE tap 4 gain control

Ethernet Tx PMA Driver Page

The Tx PMA Driver page(s) of the Speedster22i Ethernet Configuration Editor (see page 63) allow the user to set the per-lane or shared options for Tx PMA Drivers. (If per-lane settings are required, per-lane configuration must be enabled on the Ethernet PMA Settings Page (see page 67).

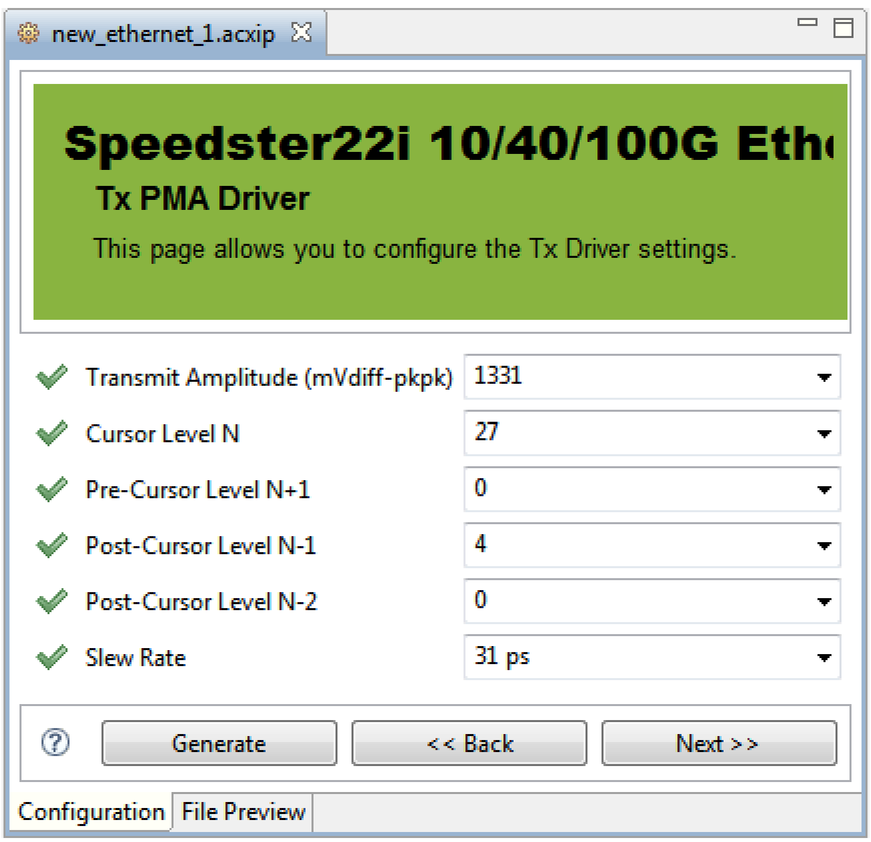


Table 20: *Ethernet Tx PMA Driver Options*

Option	Description
Transmit Amplitude (mVdiff-pkpk)	Defines the full-scale maximum swing of the driver.

Option	Description
Cursor Level N	Defines the total number of driver units allocated to the sum of the driver taps.
Pre-Cursor Level N+1	Defines the total number of driver units allocated to the first pre-cursor.
Post-Cursor Level N-1	Defines the total number of driver units allocated to the sum of the first post-cursor tap.
Post-Cursor Level N-2	Defines the total number of driver units allocated to the sum of the second post-cursor tap.
Slew Rate	Tx driver Slew Rate control.

Ethernet Link Tuning Page

This is functionally identical to the SerDes Link Tuning Page, (see page 146) and thus will not be duplicated here. See that page for more details.

Speedster22i FIFO Configuration Editor

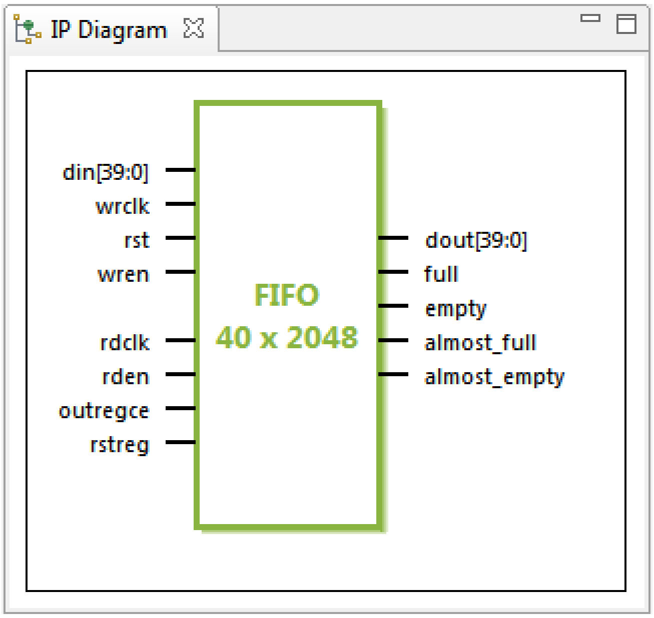
The FIFO Configuration Editor provides a graphical wizard for creating a FIFO configuration file (`.acxip`). This editor allows the user to generate the required configuration files for design with the embedded FIFOs. See Creating an IP Configuration (see page 387).

By default, the FIFO Configuration Editor is included in the IP Configuration perspective (**Window → Open Perspective → IP Configuration**).

Once the user has configured the FIFO to meet their requirements, and the FIFO Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The module diagram in the IP Diagram View (see page 196) shows the inputs and outputs of the current FIFO configuration.



FIFO Overview Page

The Overview page contains the top-level, global properties that govern the structure and base configuration of the FIFO.

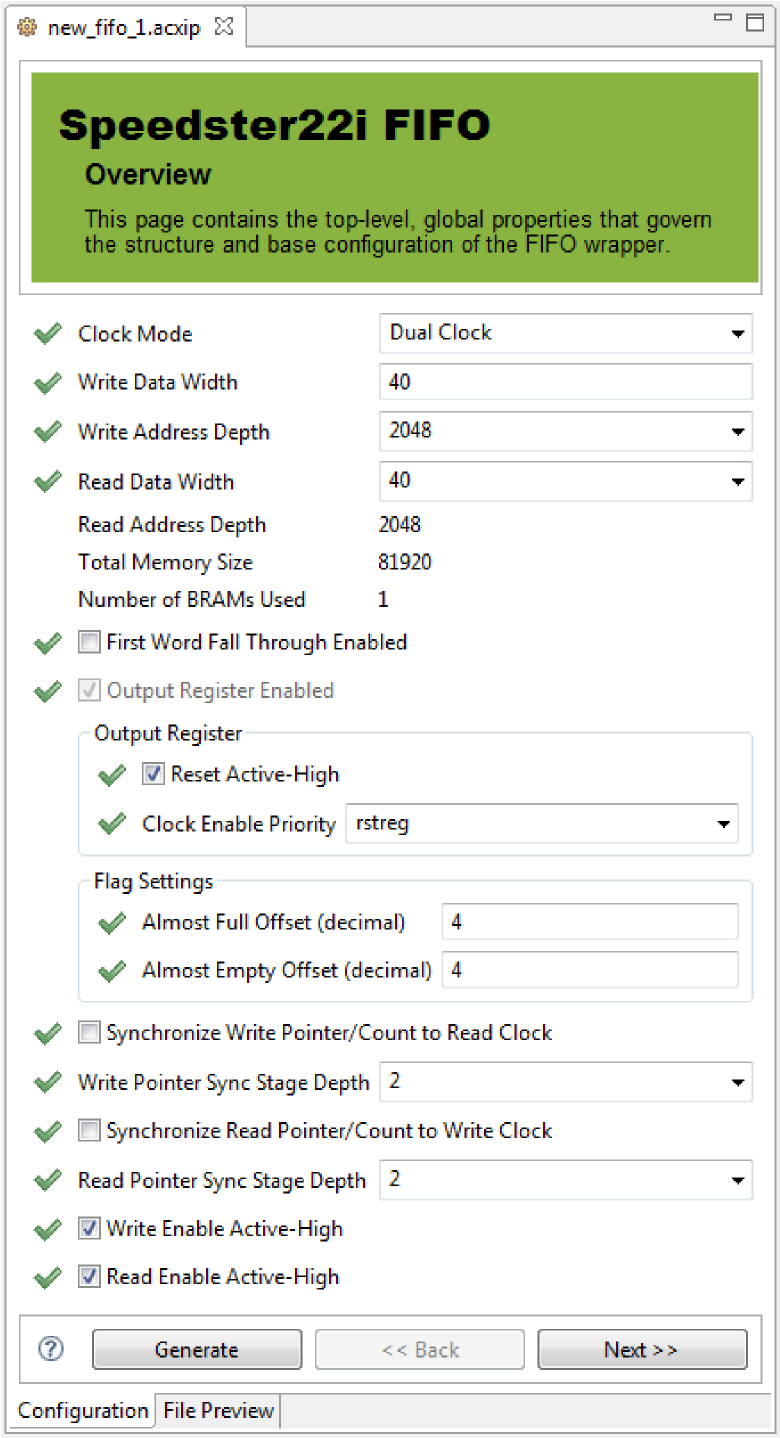

new_fifo_1.acxip
Speedster22i FIFO
Overview
This page contains the top-level, global properties that govern the structure and base configuration of the FIFO wrapper.
Clock Mode
Dual Clock
Write Data Width
40
Write Address Depth
2048
Read Data Width
40
Read Address Depth
2048
Total Memory Size
81920
Number of BRAMs Used
1
First Word Fall Through Enabled
Output Register Enabled
Output Register
Reset Active-High
Clock Enable Priority
rstreg
Flag Settings
Almost Full Offset (decimal)
4
Almost Empty Offset (decimal)
4
Synchronize Write Pointer/Count to Read Clock
Write Pointer Sync Stage Depth
2
Synchronize Read Pointer/Count to Write Clock
Read Pointer Sync Stage Depth
2
Write Enable Active-High
Read Enable Active-High
Generate
<< Back
Next >>
Configuration
File Preview

Table 21: *FIFO Editor Overview Page Options*

Option		Editable	Description
Clock Mode		Y	FIFOs can be configured in Single Clock mode to use a single clock domain for writes and reads. Single clock mode bypasses the synchronization circuitry to enable faster updates to status flags. Dual Clock mode allows two independent clocks to be used for reads and writes.
Write Data Width		Y	The FIFO write data width.
Write Address Depth		Y	The FIFO address depth is the total number of writable data words in the FIFO.
Read Data Width		Y	The FIFO read data width.
Read Address Depth			The total number of readable data words in the FIFO.
Total Memory Size			The total memory size in bits.
Number of BRAMs Used			The number of BRAMs used in the configuration.
First Word Fall Through Enabled		Y	When enabled, the first value written into the FIFO appears at the dout output without having to perform a read operation. If First Word Fall Through is disabled, the first data word written into the FIFO is available at the FIFO output one rdclk clock cycle after the first read operation. This parameter only affects the availability of the first word written into the FIFO after an empty condition. Operation of the two modes is the same after the first read operation is performed.
Output Register Enabled		Y	When the Output Register is enabled, there is an additional cycle of latency for each read operation. The Output Register is always enabled in Dual Clock mode.
Output Register			
	Reset Active High	Y	When this is enabled, the output register has an active-high synchronous reset. Otherwise, the output register reset will be active-low.
			The Clock Enable Priority controls the relationship between the *outregce* clock enable input and the *rstreg* reset input during an assertion of the *rstreg* signal on the output register. Setting the value to **rstreg** allows the output register to be set/reset at the next active edge of the *rdclk*

Option	Editable	Description
Clock Enable Priority	Y	without requiring a specific value on the *outregce* output register clock enable input. Setting the value to **regce** requires that the *outregce* output register clock enable input is active for the output register set/reset operation to occur at the next active edge of the *rdclk* .
Flag Settings		
Almost Full Offset (decimal)	Y	This defines the word depth at which the FIFO *almost_full* signal is asserted. The *almost_full* flag is asserted when there are (*afull_offset* + 1) or fewer locations available to be written in the FIFO. The *almost_full* signal is asserted when the the difference between the Write Pointer and the Read Pointer is greater than or equal to the difference between the Maximum FIFO Depth and the value of this field (*afull_offset* parameter).
Almost Empty Offset (decimal)	Y	This defines the word depth at which the FIFO *almost_empty* signal is asserted. The *almost_empty* flag is asserted when there are (*aempty_offset* - 1) or fewer words remaining in the FIFO. The *almost_empty* signal is asserted when the the difference between the Write Pointer and the Read Pointer is less than the value of this field (*aempty_offset* parameter).
Synchronize Write Pointer /Count to Read Clock	Y	When enabled, the Write Count (*wrcount*) output is synchronized to the Read Clock (*rdclk*) input. Otherwise, if left unchecked, the Write Clock output will be synchronized to the Write Clock (*wrclk*) input.
Write Pointer Sync Stage Depth	Y	The *wrptr_sync_stages* parameter defines the number of stages used in the Write Pointer Synchonizer circuit that synchronizes the Write Pointer to the *rdclk* clock domain. When the FIFO is in Dual Clock mode, the output of the synchonized Write Pointer is compared to the Read Pointer to generate the *empty* and *almost_empty* flags.
Synchronize Read Pointer /Clock to Write Clock	Y	When enabled, the Read Count (*rdcount*) output is synchronized to the Write Clock (*wrclk*) input. Otherwise, if left unchecked, the Read Clock output will be synchronized to the Read Clock (*rdclk*) input.
Read Pointer Sync Stage Depth	Y	The *rdptr_sync_stages* parameter defines the number of stages used in the Read Pointer Synchonizer circuit that synchronizes the Read Pointer to the *wrclk* clock domain. When the FIFO is in Dual Clock mode, the output of the synchonized Read Pointer is compared to the Write Pointer to generate the *empty* and *almost_empty* flags.
Write Enable Active-High	Y	When this is enabled, the write enable (*wren*) pin is an active-high. Otherwise, the write enable pin will be active-low.
Read Enable Active-High	Y	When this is enabled, the read enable (*rden*) pin is an active-high. Otherwise, the read enable pin will be active-low.

Reset Configuration Page

The Reset Configuration page contains the properties that govern the reset behavior of the FIFO.

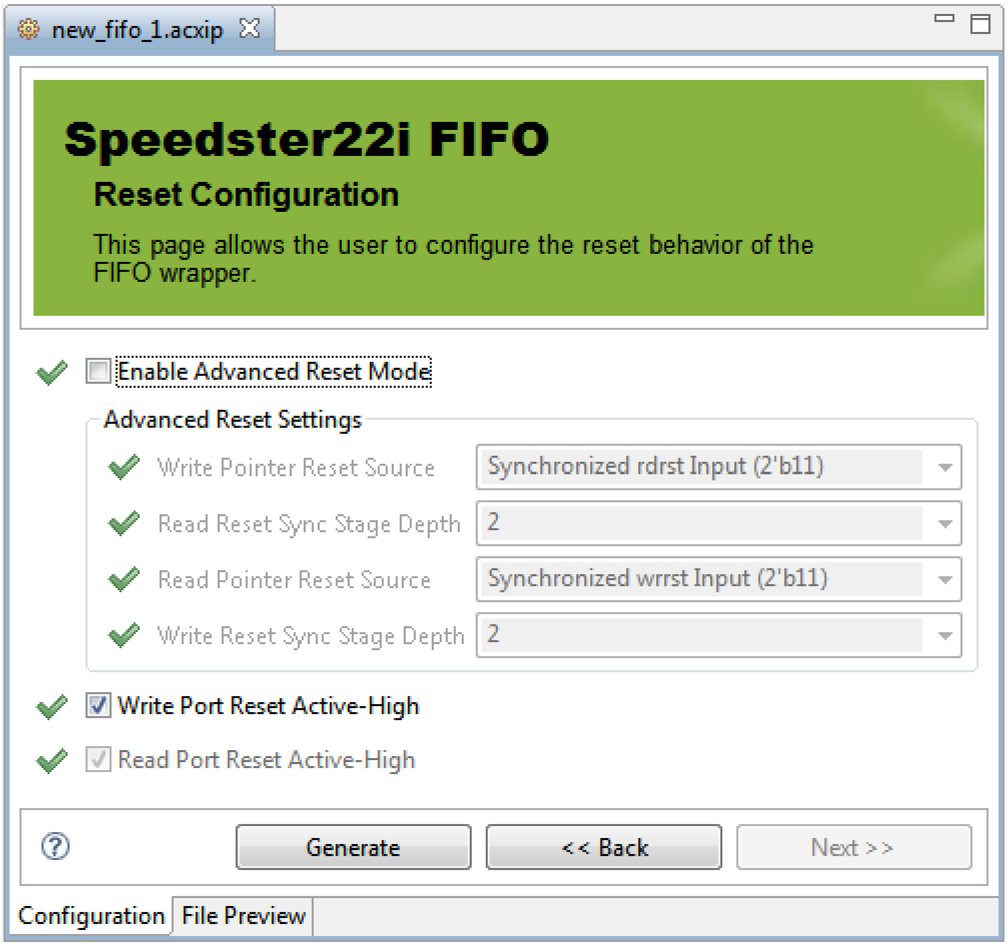


Table 22: *FIFO Editor Reset Configuration Page Options*

Option	Editable	Description
Enable Advanced Reset Mode	Y	When this is enabled, both the read and write port reset signals are exposed separately and the user can configure the advanced reset input mode and synchronization register stages. Leaving this field unchecked will configure the FIFO to use Basic Reset Mode. In Basic Reset Mode, only 1 reset signal is exposed to be shared between the read and write ports. To reset the FIFO, the user will assert the reset signal for a minimum of three clock cycles of the slower clock cycle between the wrclk and rdclk. Asserting the reset signal clears both the Write Pointer and Read Pointer, sets the empty and almost_empty flags, and clears the full and almost_full flags. The user may then release the reset signal. The user should not attempt to read or write the FIFO during, or before three cycles after the deassertion of, the reset signal.
Advanced Reset Settings		
Write Pointer		The Write Pointer Reset Source selects the reset source for the write pointer by configuring the wrrst_input_mode parameter on the FIFO. The FIFO macro provides the user with several options

Option		Editable	Description
	Reset Source	Y	to reset the FIFO either sychronously or to synchronize the reset input to the appropriate clock domain within the FIFO without the need to implement separate synchronization circuitry in the FPGA fabric.
	Read Reset Sync Stage Depth	Y	The Read Reset Sync Stage Depth defines the number of stages of registers used to synchronize the rdrst input pin to the wrclk clock domain if the rdrst signal is used by the Write Pointer Reset. The value of the rdrst_sync_stages parameter is only used if the wrrst_input_mode is set to **2'b10** or **2'b11**.
	Read Pointer Reset Source	Y	The Read Pointer Reset Source selects the reset source for the read pointer by configuring the rdrst_input_mode parameter on the FIFO. The FIFO macro provides the user with several options to reset the FIFO either sychronously or to synchronize the reset input to the appropriate clock domain within the FIFO without the need to implement separate synchronization circuitry in the FPGA fabric.
	Write Reset Sync Stage Depth	Y	The Write Reset Sync Stage Depth defines the number of stages of registers used to synchronize the wrrst input pin to the rdclk clock domain if the wrrst signal is used by the Read Pointer Reset. The value of the wrrst_sync_stages parameter is only used if the rdrst_input_mode is set to **2'b10** or **2'b11**.
Write Port Reset Active-High		Y	When this is enabled, the write port reset (wrrst) input is active-high. Otherwise, the write port reset will be active-low.
Read Port Reset Active-High		Y	When this is enabled, the read port reset (rdrst) input is active-high. Otherwise, the read port reset will be active-low.

Speedster22i Interlaken Configuration Editor

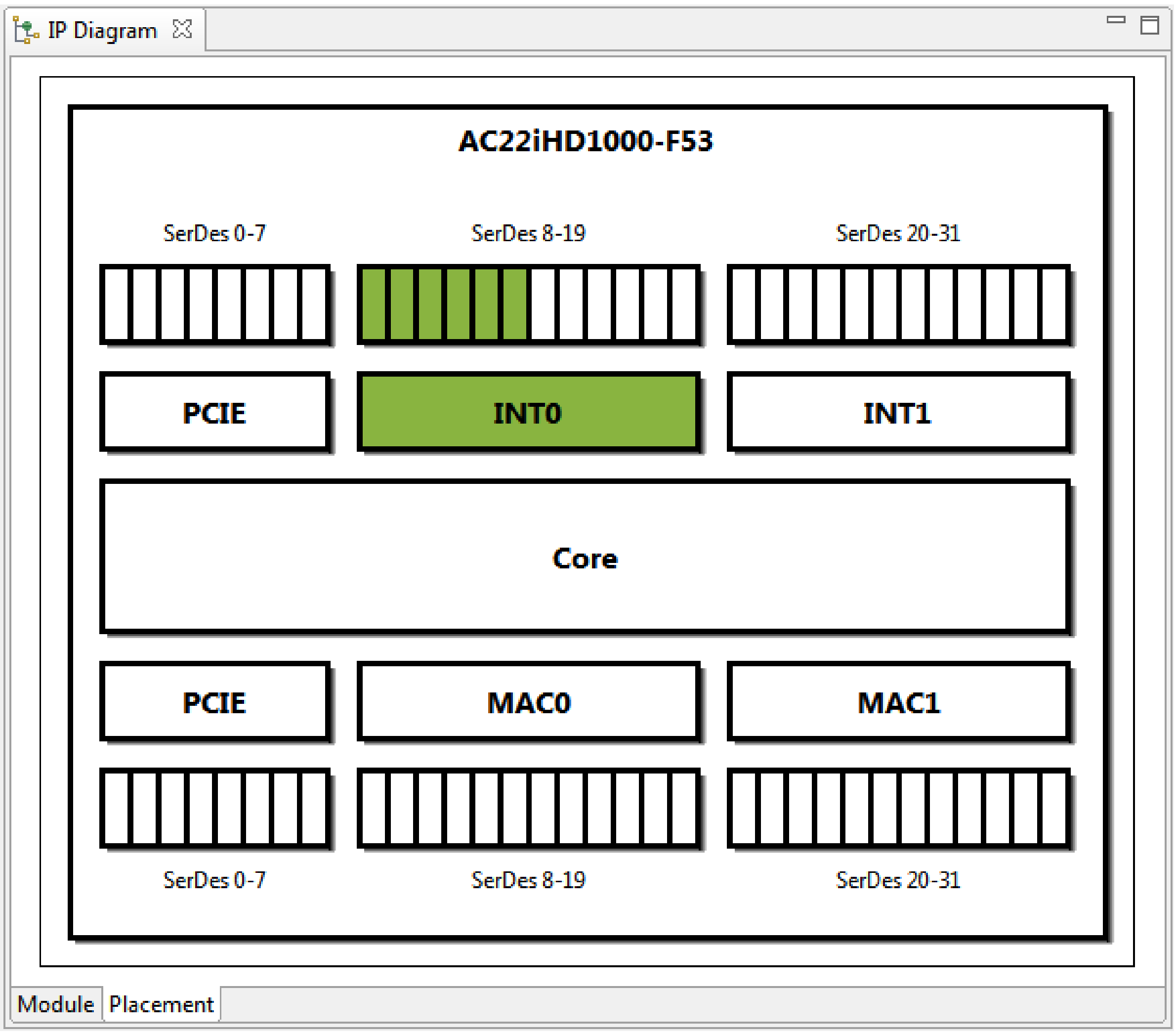
The Interlaken interface configuration editor provides a simple graphical editor used to configure the SerDes interface for Interlaken, and saves the user configuration in an Interlaken IP configuration file (`.acxip`). See Creating an IP Configuration (see page 387).

Once the user has configured the IP to meet their requirements, and the Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files. See Generating the IP Design Files (see page 389).

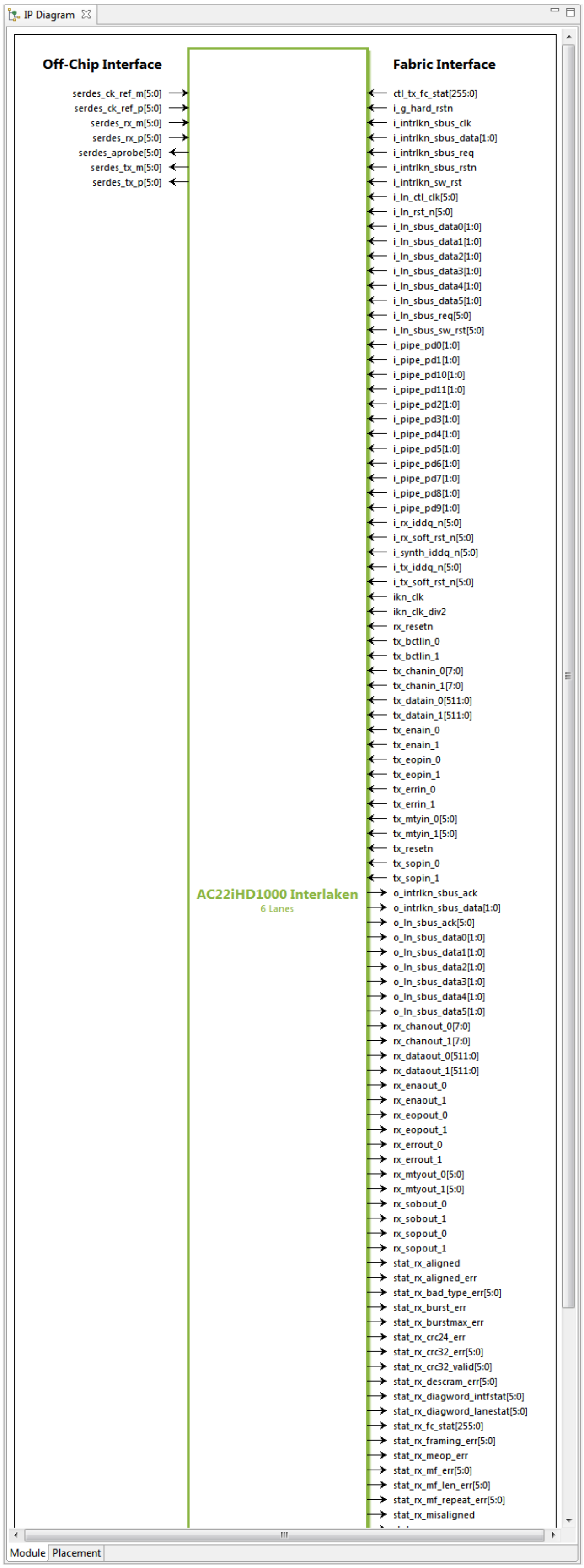
By default, the Interlaken Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**).

IP Diagrams

The IP Diagram View (see page 196)'s Placement Diagram shows the currently-selected placement of the Interlaken module according to the settings selected in the Editor.



The Module Diagram will display all the inputs and outputs of the Interlaken instance according to the current configuration.


IP Diagram
Off-Chip Interface
serdes_ck_ref_m[5:0]
serdes_ck_ref_p[5:0]
serdes_rx_m[5:0]
serdes_rx_p[5:0]
serdes_aprobe[5:0]
serdes_tx_m[5:0]
serdes_tx_p[5:0]
Fabric Interface
ctl_tx_fc_stat[255:0]
i_g_hard_rstn
i_intrlkn_sbus_clk
i_intrlkn_sbus_data[1:0]
i_intrlkn_sbus_req
i_intrlkn_sbus_rstn
i_intrlkn_sw_rst
i_ln_ctl_clk[5:0]
i_ln_rst_n[5:0]
i_ln_sbus_data0[1:0]
i_ln_sbus_data1[1:0]
i_ln_sbus_data2[1:0]
i_ln_sbus_data3[1:0]
i_ln_sbus_data4[1:0]
i_ln_sbus_data5[1:0]
i_ln_sbus_req[5:0]
i_ln_sbus_sw_rst[5:0]
i_pipe_pd0[1:0]
i_pipe_pd1[1:0]
i_pipe_pd10[1:0]
i_pipe_pd11[1:0]
i_pipe_pd2[1:0]
i_pipe_pd3[1:0]
i_pipe_pd4[1:0]
i_pipe_pd5[1:0]
i_pipe_pd6[1:0]
i_pipe_pd7[1:0]
i_pipe_pd8[1:0]
i_pipe_pd9[1:0]
i_rx_iddq_n[5:0]
i_rx_soft_rst_n[5:0]
i_synth_iddq_n[5:0]
i_tx_iddq_n[5:0]
i_tx_soft_rst_n[5:0]
ikn_clk
ikn_clk_div2
rx_resetn
tx_bctlin_0
tx_bctlin_1
tx_chanin_0[7:0]
tx_chanin_1[7:0]
tx_datain_0[511:0]
tx_datain_1[511:0]
tx_enain_0
tx_enain_1
tx_eopin_0
tx_eopin_1
tx_errin_0
tx_errin_1
tx_mtyin_0[5:0]
tx_mtyin_1[5:0]
tx_resetn
tx_sopin_0
tx_sopin_1
AC22iHD1000 Interlaken
6 Lanes
o_intrlkn_sbus_ack
o_intrlkn_sbus_data[1:0]
o_ln_sbus_ack[5:0]
o_ln_sbus_data0[1:0]
o_ln_sbus_data1[1:0]
o_ln_sbus_data2[1:0]
o_ln_sbus_data3[1:0]
o_ln_sbus_data4[1:0]
o_ln_sbus_data5[1:0]
rx_chanout_0[7:0]
rx_chanout_1[7:0]
rx_dataout_0[511:0]
rx_dataout_1[511:0]
rx_enaout_0
rx_enaout_1
rx_eopout_0
rx_eopout_1
rx_errout_0
rx_errout_1
rx_mtyout_0[5:0]
rx_mtyout_1[5:0]
rx_sobout_0
rx_sobout_1
rx_sopout_0
rx_sopout_1
stat_rx_aligned
stat_rx_aligned_err
stat_rx_bad_type_err[5:0]
stat_rx_burst_err
stat_rx_burstmax_err
stat_rx_crc24_err
stat_rx_crc32_err[5:0]
stat_rx_crc32_valid[5:0]
stat_rx_descram_err[5:0]
stat_rx_diagword_intfstat[5:0]
stat_rx_diagword_lanestat[5:0]
stat_rx_fc_stat[255:0]
stat_rx_framing_err[5:0]
stat_rx_meop_err
stat_rx_mf_err[5:0]
stat_rx_mf_len_err[5:0]
stat_rx_mf_repeat_err[5:0]
stat_rx_misaligned
Module
Placement

Interlaken Overview Page

The Overview page contains the top-level, global options that govern the structure and base configuration of the Interlaken interface wrapper.

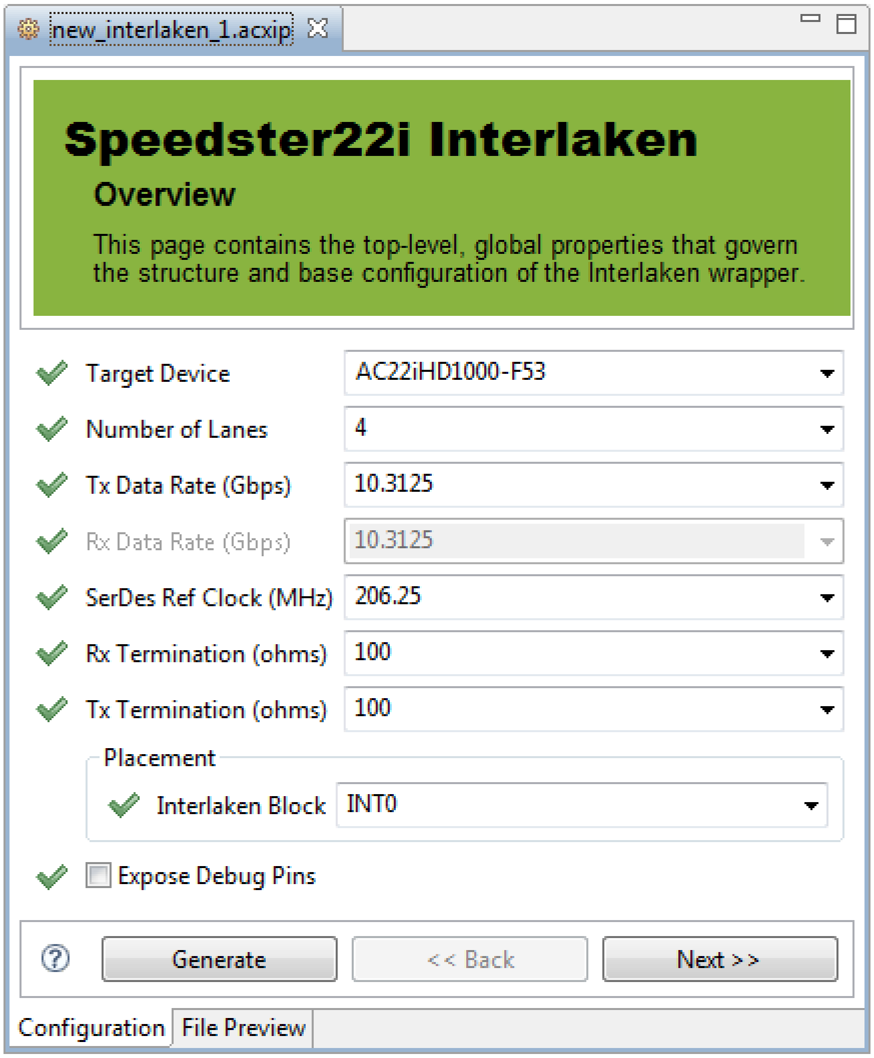


Table 23: *Interlaken Editor Overview Page Options*

Option	Editable	Description
Target Device	Y	The Target Device allows you to select from any compatible Speedster22i devices for placing SerDes Lanes.
Number of Lanes	Y	Sets the number of lanes to be exposed in the RTL wrapper.
Tx Data Rate (Gbps)	Y	Desired transmit-side data rate for the SerDes. The Rx Data Rate will match this value.

Option	Editable	Description
Rx Data Rate (Gbps)		The receive-side data rate for the SerDes, which is always identical to the transmit-side data rate.
SerDes Ref Clock (MHz)	Y	Reference clock frequency to be provided to the SerDes.
Rx Termination (ohms)	Y	Rx lane calibration impedance setting.
Tx Termination (ohms)	Y	Tx lane calibration impedance setting.
Placement		
Interlaken Block	Y	Selects which Interlaken site will be used for placement. (See the Placement IP Diagram.)
Expose Debug Pins	Y	Expose Interlaken and SerDes debug signals on generated wrapper

Interlaken PMA Settings Page

This page allows the selection of lane-based or uniform PMA, as well as Ref Clock Termination values.

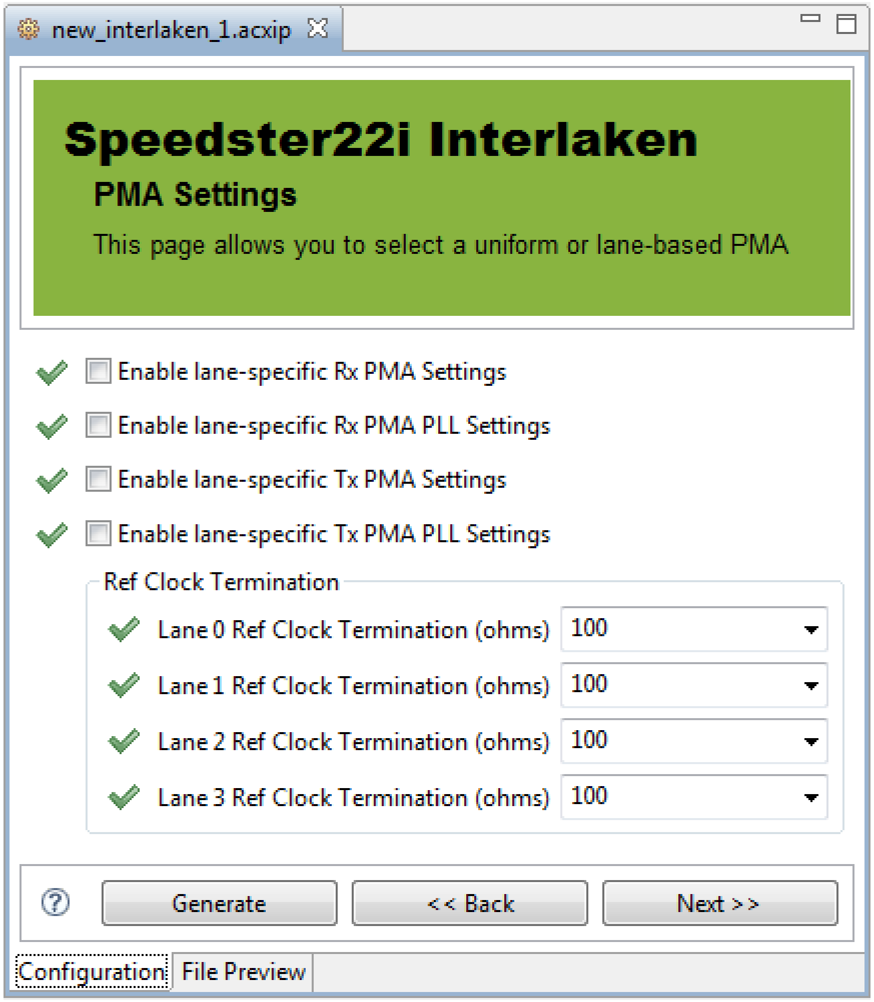


Figure 5: Screenshot for a four-lane configuration

Option	Description
Enable lane-specific Rx PMA Settings	If enabled, instead of a single page configuring the Rx PMA settings for all lanes, each lane will have its own page.
Enable lane-specific Rx PMA PLL Settings	If enabled, instead of a single page configuring the Rx PMA PLL settings for all lanes, each lane will have its own page.
Enable lane-specific Tx PMA Settings	If enabled, instead of a single page configuring the Tx PMA settings for all lanes, each lane will have its own page.
Enable lane-specific Tx PMA PLL Settings	If enabled, instead of a single page configuring the Tx PMA PLL settings for all lanes, each lane will have its own page.
Ref Clock Termination	

Option	Description
Lane [*0..N*] Ref Clock Termination (ohms)	Ref clock impedance setting. The number of lanes shown (*N*) will vary with the number of lanes active in the configuration. Ref clocks are tied to pairs of lanes in AC22iHD1000.

Rx PMA Equalization Page

This page allows the customization of the PMA equalization settings of the Interlaken wrapper.

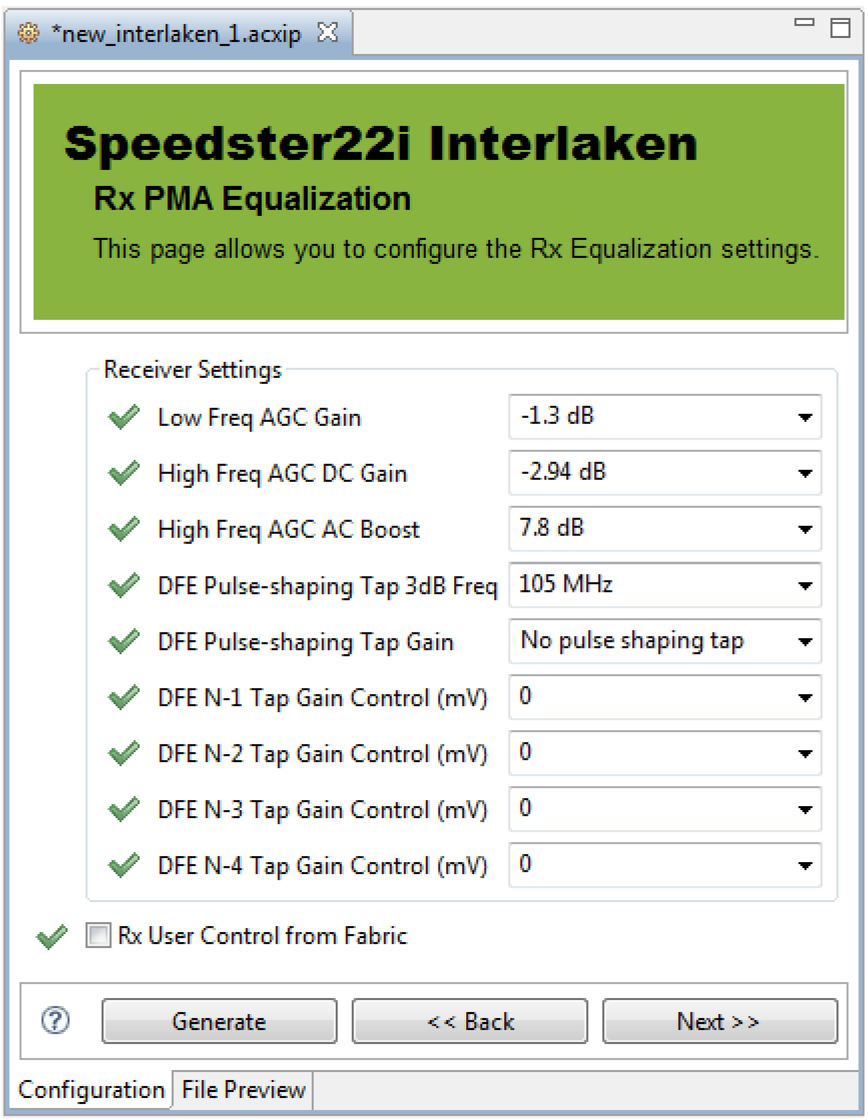


Table 24: *Rx PMA Equalization Page Options*

Options		Description
Receiver Settings		
	Low Freq AGC Gain	Low frequency AGC Gain
	High Freq AGC DC Gain	Rx AGC high frequency DC gain
	High Freq AGC AC Boost	Rx AGC high frequency AC boost
	DFE Pulse-shaping Tap 3dB Freq	DFE pulse-shaping tap 3dB frequency
	DFE Pulse-shaping Tab Gain	DFE pulse-shaping tap gain
	DFE N1 Tap Gain Control (mV)	DFE tap 1 gain control
	DFE N2 Tap Gain Control (mV)	DFE tap 2 gain control
	DFE N3 Tap Gain Control (mV)	DFE tap 3 gain control
	DFE N4 Tap Gain Control (mV)	DFE tap 4 gain control
Rx User Control from Fabric		Control Rx PMA settings from user logic with `chX_i_pma_rxeqlut[32:0]` and `chX_i_pma_rxeqlut_str`.

Rx PMA PLL Page

This page allows customization of the Rx PLL settings of the Interlaken wrapper.

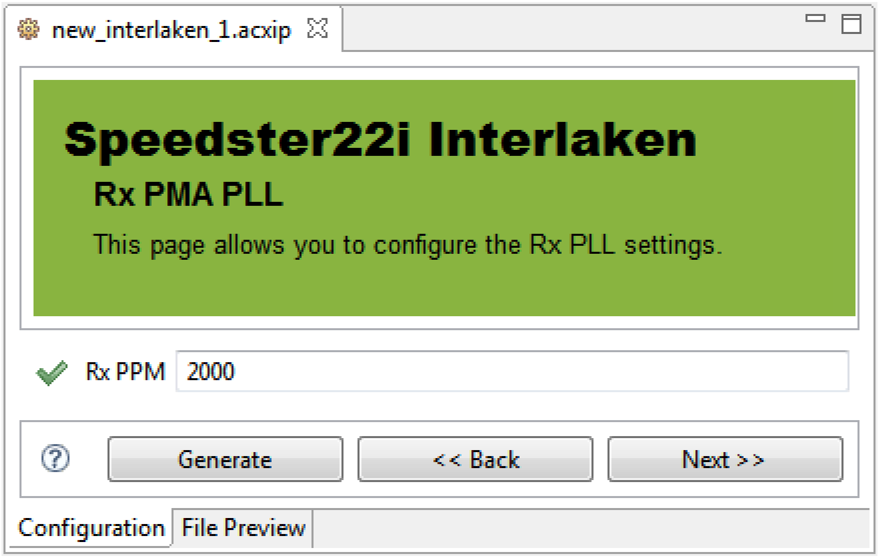


Table 25: *Rx PMA PLL Page Options*

Option	Description
Rx PPM	When Rx PLL is within this PPM range PMA will consider Tx PLL to be locked

Tx PMA Driver Page

This page allows the customization of the Tx PMA Driver settings for the Interlaken wrapper.

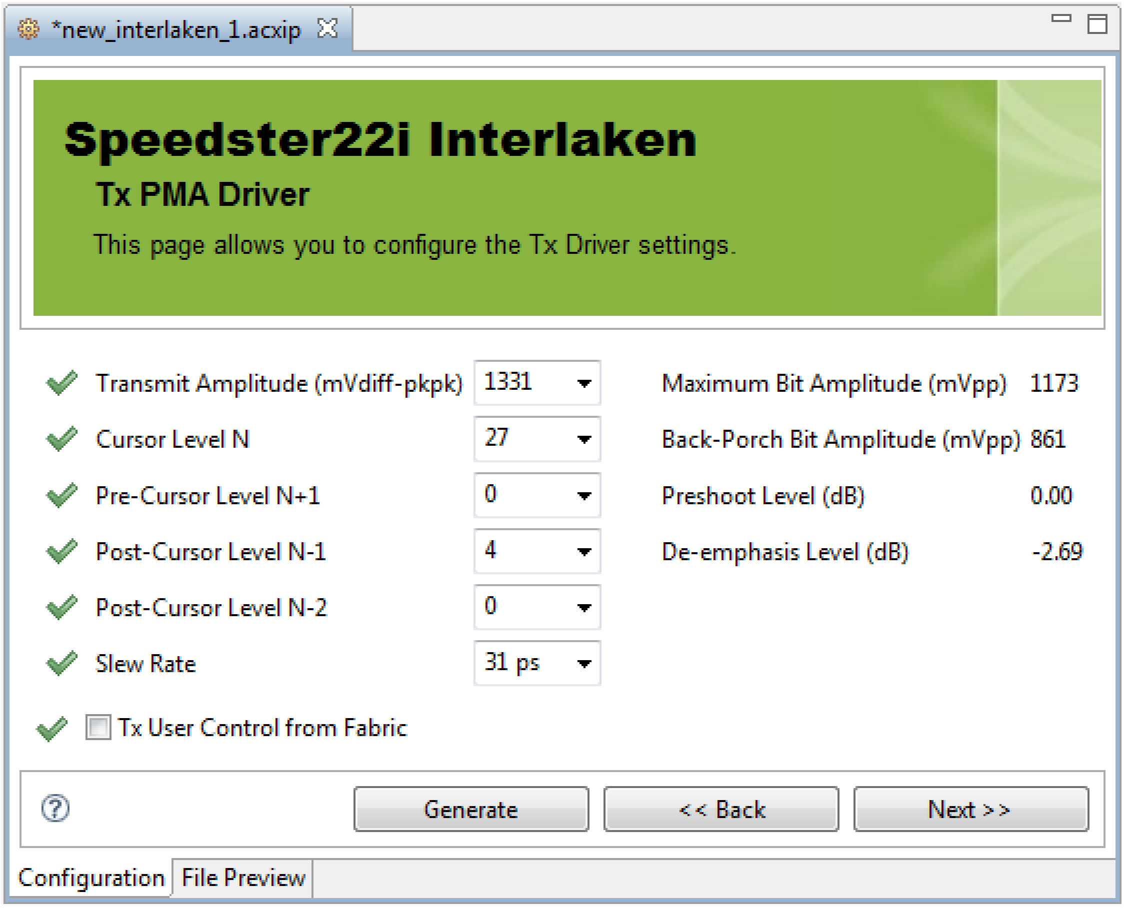


Table 26: *Tx PMA Driver Page Options*

Option	Editable	Description
Transmit Amplitude (mVdiff-pkpk)	✓	Defines the full-scale maximum swing of the driver
Cursor Level N	✓	Defines the total number of driver units allocated to the sum of the driver taps.
Pre-Cursor Level N+1	✓	Defines the total number of driver units allocated to the first pre-cursor
Post-Cursor Level N-1	✓	Defines the total number of driver units allocated to the sum of the first post-cursor tap.
Post-Cursor Level N-2	✓	Defines the total number of driver units allocated to the sum of the second post-cursor tap.
Slew Rate	✓	Tx driver Slew Rate control.
Tx User Control from Fabric	✓	Control PMA Transmit de-emphasis from fabric

Option	Editable	Description
Maximum Bit Amplitude (mVpp)		(DC Amp N) + (DC Amp N+1) + (DC Amp N-1) + (DC Amp N-2)
Back-Porch Bit Amplitude (mVpp)		(DC Amp N) - (DC Amp N+1) - (DC Amp N-1) - (DC Amp N-2)
Preshoot Level (dB)		20 × $\log_{10}$(((DC Amp N) - (DC Amp N+1) - (DC Amp N-1)) ÷ ((DC Amp N) + (DC Amp N+1) - (DC Amp N-1)))
De-emphasis Level (dB)		20 × $\log_{10}$(((DC Amp N) - (DC Amp N+1) - (DC Amp N-1)) ÷ ((DC Amp N) - (DC Amp N+1) + (DC Amp N-1)))

Tx PMA PLL Page

This page allows the customization of the Tx PMA PLL settings for the Interlaken wrapper.

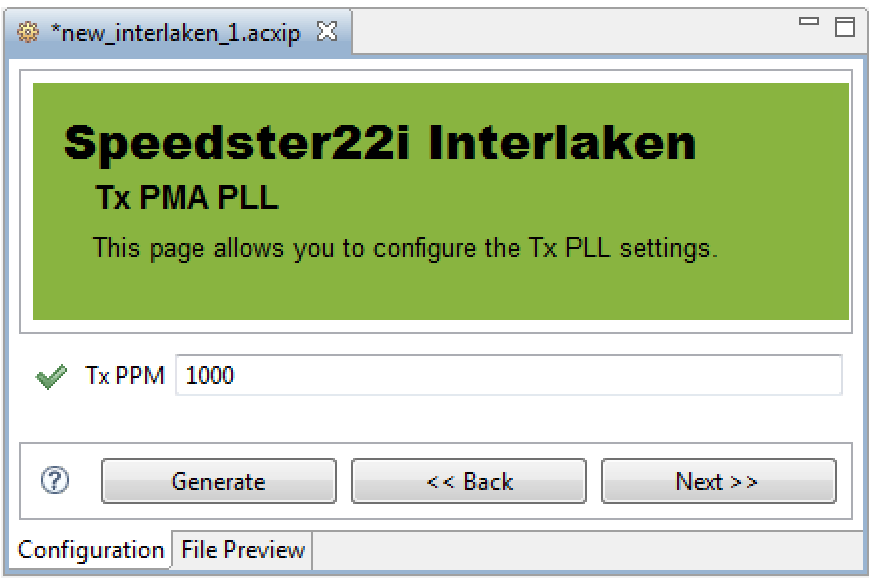


Table 27: *Tx PMA PLL Settings Page Options*

Option	Description
Tx PPM	When Tx PLL is within this PPM range PMA will consider Tx PLL to be locked

Interrupt Settings Page

This page allows the customization of the Interrupts for the Interlaken wrapper.

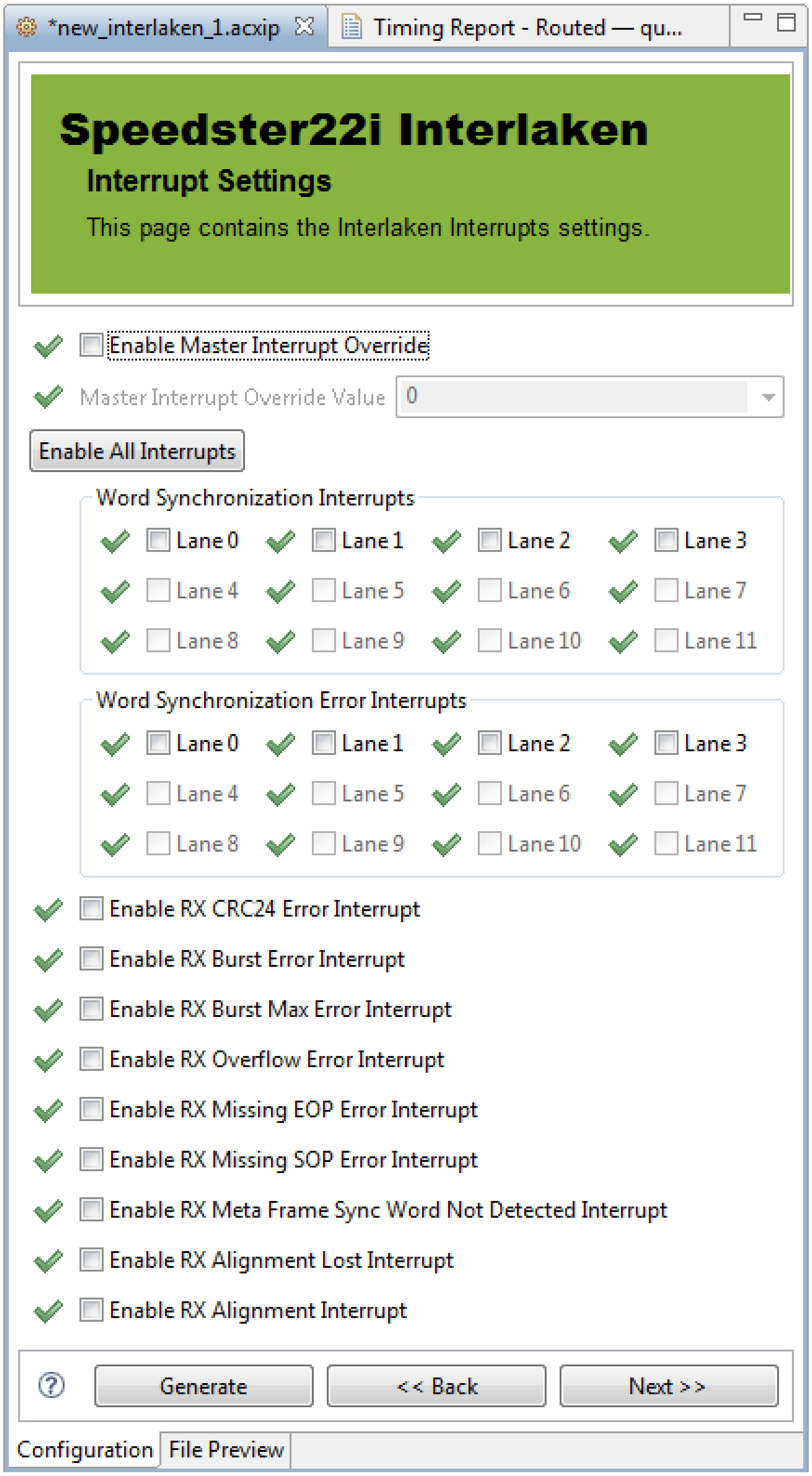


Figure 6: Screenshot for a four-lane configuration

Table 28: *Interrupt Settings Page Options*

Option		Description
Enable Master Interrupt Override		Allows the use of the Master Interrupt Override.
Master Interrupt Override Value		If enabled, this value is transmitted to the INT signal.
[Enable/Disable] All Interrupts		When pressed, enables (or disables) the checkboxes for all available interrupts
Word Synchronization Interrupts		
	Lane 0	Enables the word synchronization interrupt for lane 0.
	…	…
	Lane 11	Enables the word synchronization interrupt for lane 11.
Work Synchronization Error Interrupts		
	Lane 0	Enables the word synchronization error interrupt for lane 0.
	…	…
	Lane 11	Enables the word synchronization error interrupt for lane 11.
Enable RX CRC24 Error Interrupt		
Enable RX Burst Error Interrupt		
Enable RX Burst Max Error Interrupt		
Enable RX Overflow Error Interrupt		
Enable RX Missing EOP Error Interrupt		
Enable RX Missing SOP Error Interrupt		
Enable RX Meta Frame Sync Word Not Detected Interrupt		
Enable RX Alignment Lost Interrupt		
Enable RX Alignment Interrupt		If enabled, an interrupt is raised when all the Rx lanes are aligned.

Interlaken Tx Settings Page

This page contains the Interlaken Transmit-side settings for the Interlaken wrapper.

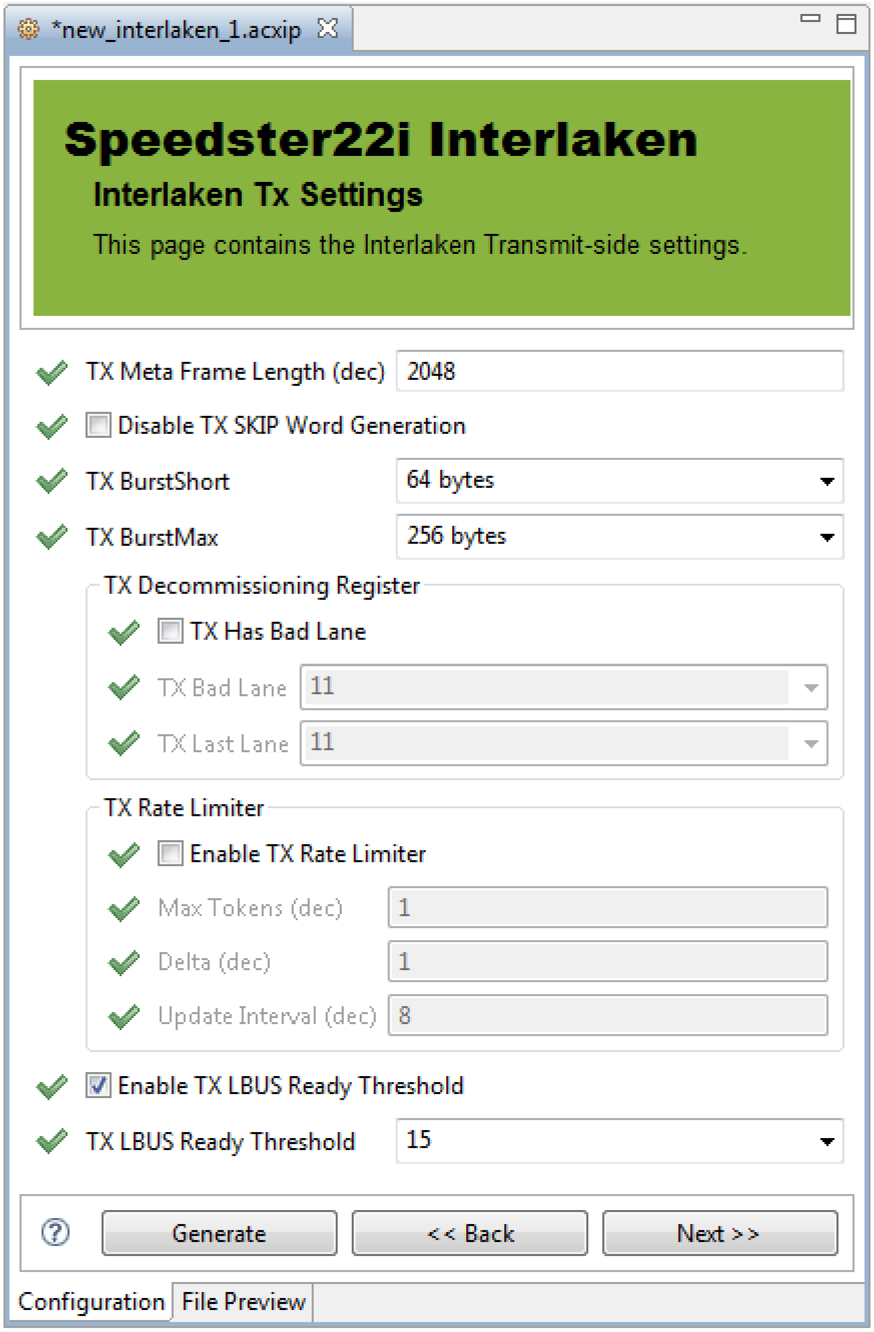


Table 29: *Interlaken Tx Settings Options*

Option	Description
TX Meta Frame Length (dec)	TX Meta Frame Length Register: This input should be set to the desired length. The parameter value generated from the GUI will be (GUI input value - 1). Thus for a Meta Frame of 2048, a value of 2048 should be used, and the Verilog parameter will be set to 2047. This input is specified in terms of the number of words or cycles minus one. For example, if set to 2048, then a Metaframe sync word is sent every 2048 word transfers on every lane. See section 5.4.3 of Interlaken spec 1.1.
Disable TX SKIP Word Generation	Disables the generation of a skip word after the scrambler state word.
TX BurstShort	Specifies the minimum spacing between Burst Control Words.
TX BurstMax	Specifies the maximum number of Data Words between Burst Control Words. See section 5.3.2 of Interlaken spec 1.1.
TX Decommissioning Register	
TX Has Bad Lane	
TX Bad Lane	
TX Last Lane	
TX Rate Limiter	
Enable TX Rate Limiter	
Max Tokens (dec)	Specifies the maximum number of tokens in the bucket in terms of bytes (a value of 1 means 1 byte). The number of tokens in the bucket will never exceed this value. This value must be at least **TX BurstMax**. (For example, if **TX BurstMax** is set for 256 bytes, then this value should be at least 256). This value should not be changed in a running design when the TX Rate Limiter is enabled. **Tip** The rates closest to the expected rates have been observed to be when this field is set to a value between 1 and 2 times the value of **TX BurstMax**.

Option	Description
Delta (dec)	Specifies how many tokens are to be added to the token bucket after each interval. This value must be greater than 0. This value should not be changed in a running design when the TX Rate Limiter is enabled.
Update Interval (dec)	Specifies the interval, in Local bus clock cycles, that the token bucket bucket will be updated. It is recommended that this value be greater than or equal to 8. This value should not be changed in a running design when the TX Rate Limiter is enabled.
Enable TX LBUS Ready Threshold	
TX LBUS Ready Threshold	

Interlaken Rx Settings Page

This page configures the Interlaken Receive-side settings for the Interlaken wrapper.

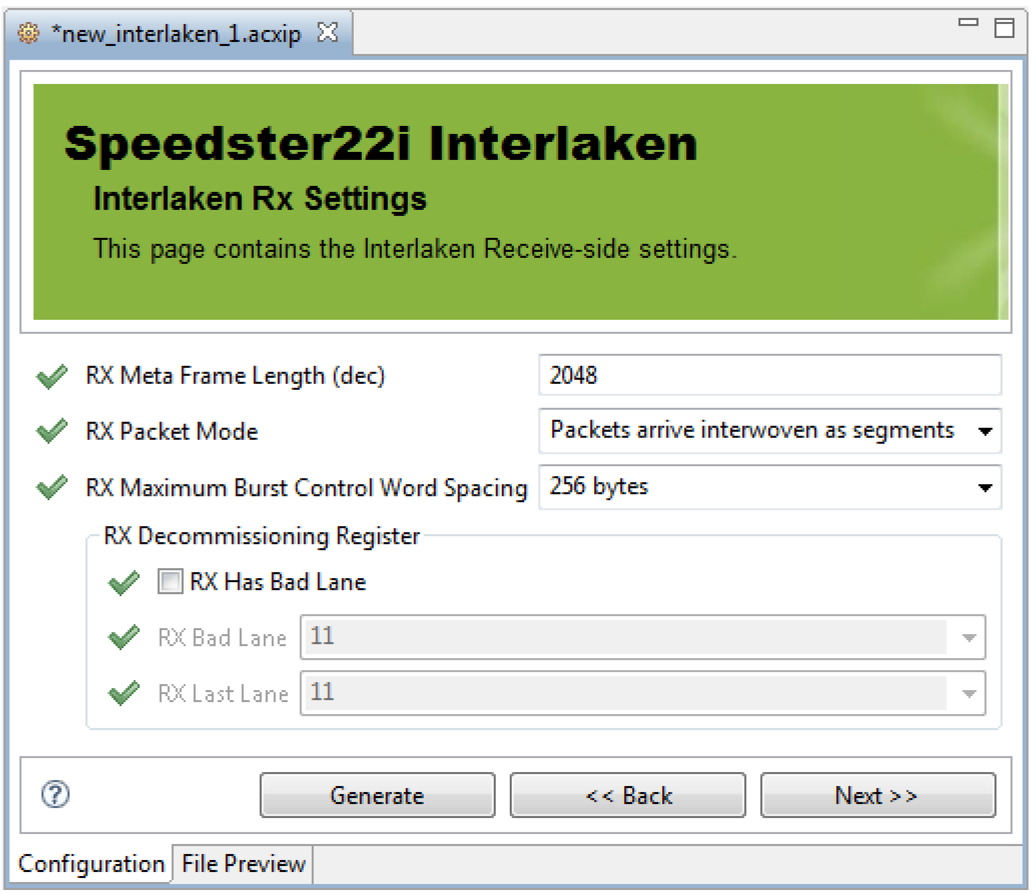


Table 30: ***Interlaken Rx Settings Page Options***

Option	Description
RX Meta Frame Length (dec)	This input should be set to the desired length. The parameter value generated from the GUI will be (GUI input value - 1). Thus for a Meta Frame of 2048, a value of 2048 should be used, and the Verilog parameter will be set to 2047. For example, if set to 2048, then a Metaframe sync word is sent every 2048 word transfers on every lane. See section 5.4.3 of Interlaken spec 1.1.
RX Packet Mode	Changes the way the error handler reports errors. Either packets are expected to arrive interwoven as segments, or packets are expected to arrive as complete packets. This setting ensures that packets delivered to the Local bus had the appropriate SOP and EOP pairing.
RX Maximum Burst Control Word Spacing	Specifies the maximum number of Data Words between Burst Control Words expected by the RX.
RX Decommissioning Register	
RX Has Bad Lane	
RX Bad Lane	
RX Last Lane	

Interlaken Link Tuning Page

This is functionally identical to the SerDes Link Tuning Page, (see page 146) and thus will not be duplicated here. See that page for more details.

Speedster22i LRAM Configuration Editor

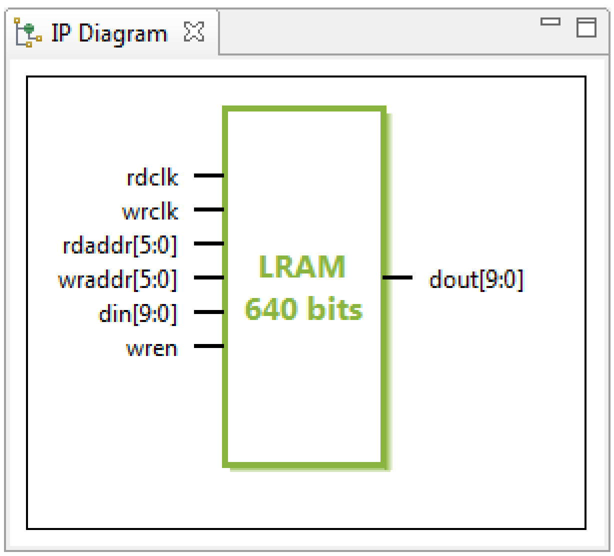
The LRAM Configuration Editor provides a simplified graphical wizard for creating an LRAM wrapper IP configuration file (.acxip). This editor allows the user to generate the required configuration files for design with the embedded LRAMs. See Creating an IP Configuration (see page 387).

By default, the LRAM Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**). The LRAM configuration information fits into a single page.

Once the user has configured the LRAM wrapper to meet their requirements, and the LRAM Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The IP Diagram View (see page 196) for the LRAM shows live information about the current configuration in the Editor, including the total memory size and which inputs and outputs are currently active. Additionally, relevant configuration errors will be shown with a red background, and configuration warnings will be shown with a yellow background (these are the default IP Diagram colors, and may be modified in the Preferences).



1. Overview Page (see page 93)

LRAM Overview Page

The Overview page contains all the properties that govern the structure and configuration of the LRAM wrapper.

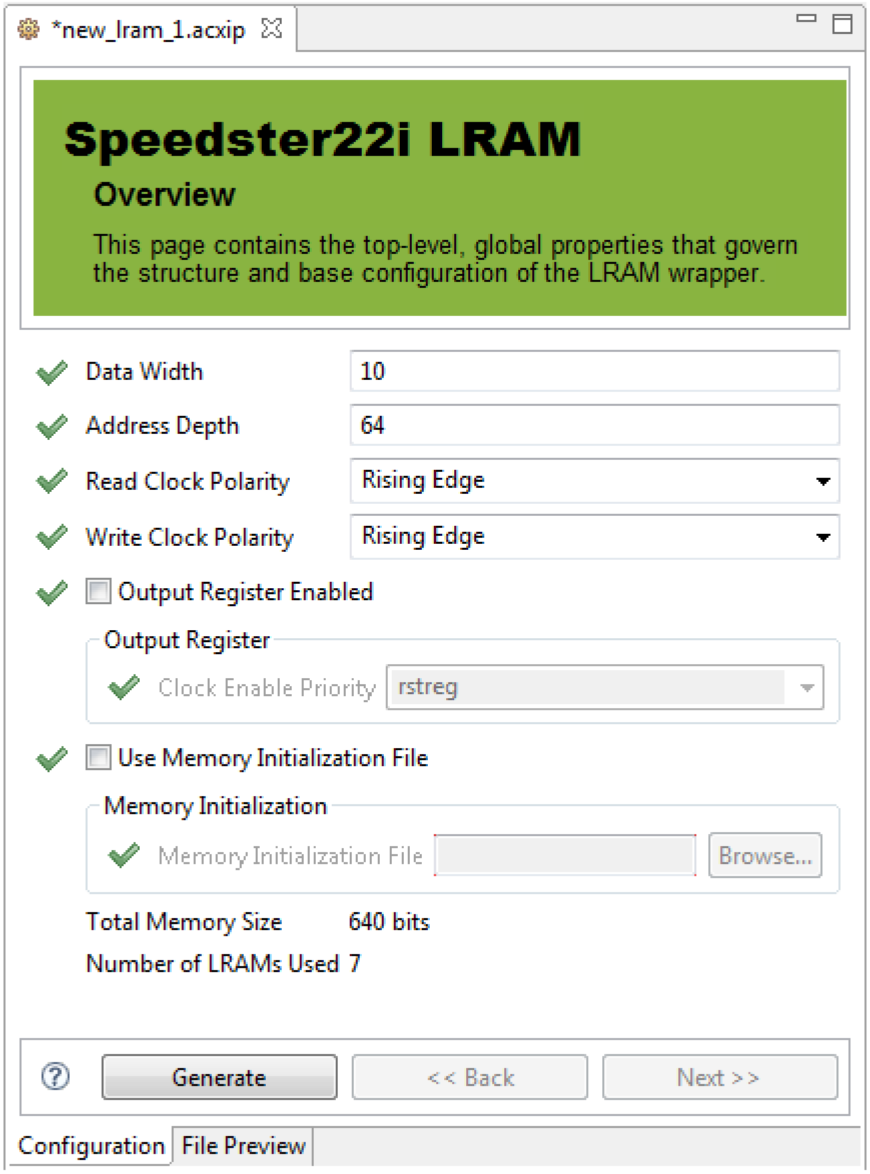


Table 31: *LRAM Editor Overview Page Options*

Option	Editable	Description
Data Width	Y	Data width of read and write ports.
Address Depth	Y	Desired address depth of the LRAM in words.
Read Clock Polarity	Y	The read port clock polarity can be set to use either rising edge assignment or falling edge assignment.
Write Clock Polarity	Y	The write port clock polarity can be set to use either rising edge assignment or falling edge assignment.

Option	Editable	Description
Output Register Enabled	Y	When the Output Register is enabled, there is an additional cycle of latency for each read operation.
Output Register		
Clock Enable Priority	Y	The Clock Enable Priority defines the priority of the `outregce` clock enable input relative to the `rstreg` reset input during an assertion of the `rstreg` signal on the read port output register. The value **rstreg** allows the Port A output register to be set/reset at the next active edge of the read port clock without requiring a specific value on the `outregce` output register clock enable input. The value **regce** requires that the `outregce` output register clock enable input is high for the output register set/reset operation to occur at the next active edge of the read port clock.
Use Memory Initialization File	Y	Enable the use of a Memory Initialization File.
Memory Initialization		
Memory Initialization File	Y	Path to initialization file whose data is “Data Width” wide and “Address Depth” deep. The memory initialization file should be in hexadecimal.
Total Memory Size		Reports the total memory size for the currently configuration, in bits.
Number of LRAMS Used		Reports the total number of LRAMs which will be instantiated to support the current configuration.

Speedster22i LRAM FIFO Configuration Editor

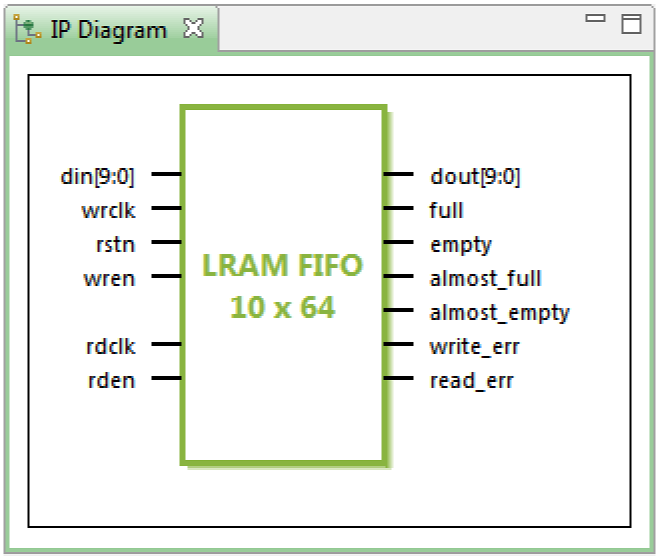
The LRAM FIFO Configuration Editor provides a graphical wizard for creating an LRAM FIFO configuration file (`.acxip`). This editor allows the user to generate the required configuration files for design with the embedded LRAMs. See Creating an IP Configuration (see page 387).

By default, the LRAM FIFO Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**).

Once the user has configured the LRAM FIFO to meet their requirements, and the LRAM FIFO Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The module diagram in the IP Diagram View (see page 196) shows the inputs and outputs of the current LRAM FIFO configuration.



1. LRAM FIFO Overview Page (see page 96)

LRAM FIFO Overview Page

The Overview page contains the top-level, global properties that govern the structure and base configuration of the LRAM FIFO.

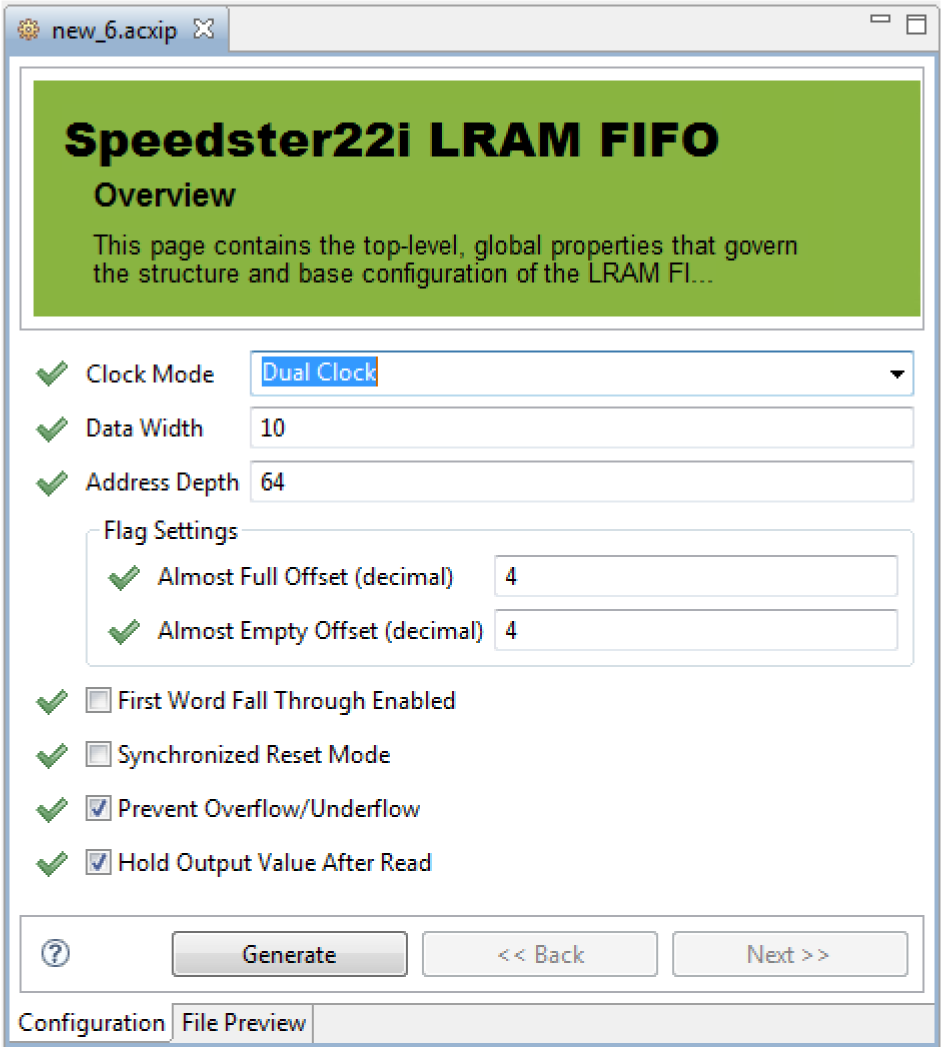


Table 32: *LRAM FIFO Editor Overview Page Options*

Option	Description
Clock Mode	FIFOs can be configured in Single Clock mode to use a single clock domain for writes and reads. Single clock mode bypasses the synchronization circuitry to enable faster updates to status flags. Dual Clock mode allows two independent clocks to be used for reads and writes.
Data Width	The FIFO read and write port data width.
Address Depth	The FIFO address depth is the total number of writable data words in the FIFO.
Flag Settings	
	This defines the word depth at which the FIFO `almost_full` signal is asserted. The `almost_full` flag is asserted when there are (afull_offset + 1) or fewer locations available to be written in the FIFO. The

Option	Description
Almost Full Offset (decimal)	`almost_full` signal is asserted when the the difference between the Write Pointer and the Read Pointer is greater than or equal to the difference between the Maximum FIFO Depth and the value of this field (afull_offset parameter).
Almost Empty Offset (decimal)	This defines the word depth at which the FIFO `almost_empty` signal is asserted. The `almost_empty` flag is asserted when there are (aempty_offset - 1) or fewer words remaining in the FIFO. The `almost_empty` signal is asserted when the the difference between the Write Pointer and the Read Pointer is less than the value of this field (aempty_offset parameter).
First Word Fall Through Enabled	When enabled, the first value written into the FIFO appears at the `dout` output without having to perform a read operation. If First Word Fall Through is disabled, the first data word written into the FIFO is available at the FIFO output one `rdclk` clock cycle after the first read operation. This parameter only affects the availability of the first word written into the FIFO after an empty condition. Operation of the two modes is the same after the first read operation is performed.
Synchronized Reset Mode	When this is disabled, both the read and write pointers resets utilize the Reset Synchronizer circuitry. When this option is enabled the `rstn` input must be synchronous to the `wrclk` / `rdclk` clock driving the FIFO.
Prevent Overflow /Underflow	Enabling this option allows the user to read from the FIFO when the FIFO is empty and write to the FIFO when it is full. Disabling this safety check will allow the FIFO to run faster. Details can be found in the user macro guide.
Hold Output Value After Read	When this is enabled, the read output holds its value until the next read. When this option is disabled the read output data is valid for 1 clock cycle after the read, and then becomes invalid. This gives a slight performance advantage in the circuit. Only disable this option if your circuit can reliably pull data from the output within 1 clock cycle of the read.

Speedster22i PCI Express Configuration Editor

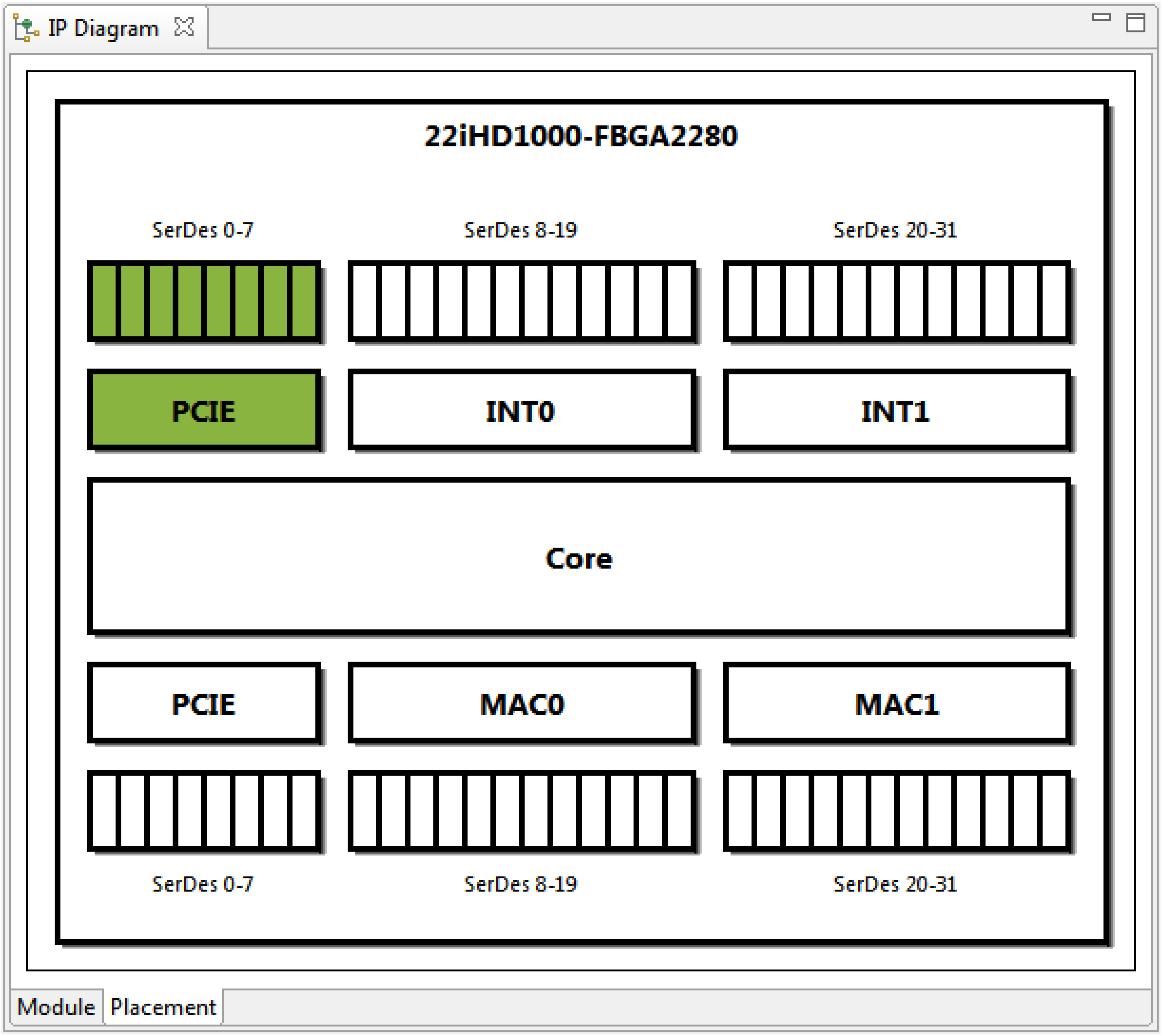
The PCI Express (PCIe) interface configuration editor provides a simple graphical editor used to configure the SerDes interface for PCIe, and saves the user configuration in a PCIe IP configuration file (.acxip). See Creating an IP Configuration (see page 387).

Once the user has configured the IP to meet their requirements, and the Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

By default, the PCI Express Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**).

IP Diagrams

The IP Diagram View (see page 196)'s Placement Diagram shows the currently-selected placement of the PCIe module according to the settings selected in the Editor.



The Module Diagram will display all the inputs and outputs of the PCIe instance according to the current configuration.


IP Diagram
Off-Chip Interface
Fabric Interface
pcie_refclk_n[7:0]
pcie_refclk_p[7:0]
rx_n[7:0]
rx_p[7:0]
tx_n[7:0]
tx_p[7:0]
c2s_aclk[1:0]
c2s_arready[1:0]
c2s_fifo_addr_n[1:0]
c2s_rdata[255:0]
c2s_rlast[1:0]
c2s_rresp[3:0]
c2s_ruserstrb[31:0]
c2s_rvalid[1:0]
core_cfg_exp_addr[9:0]
core_cfg_exp_rd_en
core_cfg_exp_wr_be[3:0]
core_cfg_exp_wr_data[31:0]
core_cfg_exp_wr_en
i_sbus_clk
i_sbus_data[1:0]
i_sbus_req
i_sbus_sw_rst
i_serdes_sbus_data[15:0]
i_serdes_sbus_req[7:0]
m_aclk
m_araddr[15:0]
m_areset_n
m_arvalid
m_awaddr[15:0]
m_awvalid
m_bready
m_rready
m_wdata[31:0]
m_wstrb[3:0]
m_wvalid
mgmt_transactions_pending
perst_n
pm_d3cold_exit
pm_d3cold_n_pme_assert
pm_d3cold_pme_asserted
pm_d3cold_restore[2:0]
pm_l2_enter_ack
s2c_aclk[1:0]
s2c_awready[1:0]
s2c_bresp[3:0]
s2c_bvalid[1:0]
s2c_fifo_addr_n[1:0]
s2c_wready[1:0]
t_aclk
t_areset_n
t_arready
t_awready
t_bresp[1:0]
t_bvalid
t_rdata[127:0]
t_rlast
t_rresp[1:0]
t_rvalid
t_wready
user_interrupt
c2s_aclk_out[1:0]
c2s_araddr[71:0]
c2s_areset_n[1:0]
c2s_arlen[7:0]
c2s_arsize[5:0]
c2s_arvalid[1:0]
c2s_rready[1:0]
c2s_ruserafull[1:0]
clk_out
core_cfg_exp_rd_data[31:0]
core_cfg_exp_rd_val
m_aclk_out
m_arready
m_awready
m_bresp[1:0]
m_bvalid
m_interrupt[4:0]
m_rdata[31:0]
m_rresp[1:0]
m_rvalid
m_wready
mgmt_cfg_id[15:0]
mgmt_dl_link_up_o
mgmt_pl_link_up_o
mgmt_rp_leg_int_o[3:0]
o_sbus_ack
o_sbus_data[1:0]
o_serdes_sbus_ack[7:0]
o_serdes_sbus_data[15:0]
pm_d3cold_exit_ack
pm_l1_enter
pm_l1_exit
pm_l2_enter
pm_l2_exit
pm_l2_store[2:0]
pm_power_state[1:0]
s2c_aclk_out[1:0]
s2c_areset_n[1:0]
s2c_awaddr[71:0]
s2c_awlen[7:0]
s2c_awsize[5:0]
s2c_awusereop[1:0]
22iHD1000 PCI Express
PCI Express Gen 3
Module
Placement

PCIE Overview Page

The Overview page contains all the options that govern the structure and configuration of the PCI Express interface.

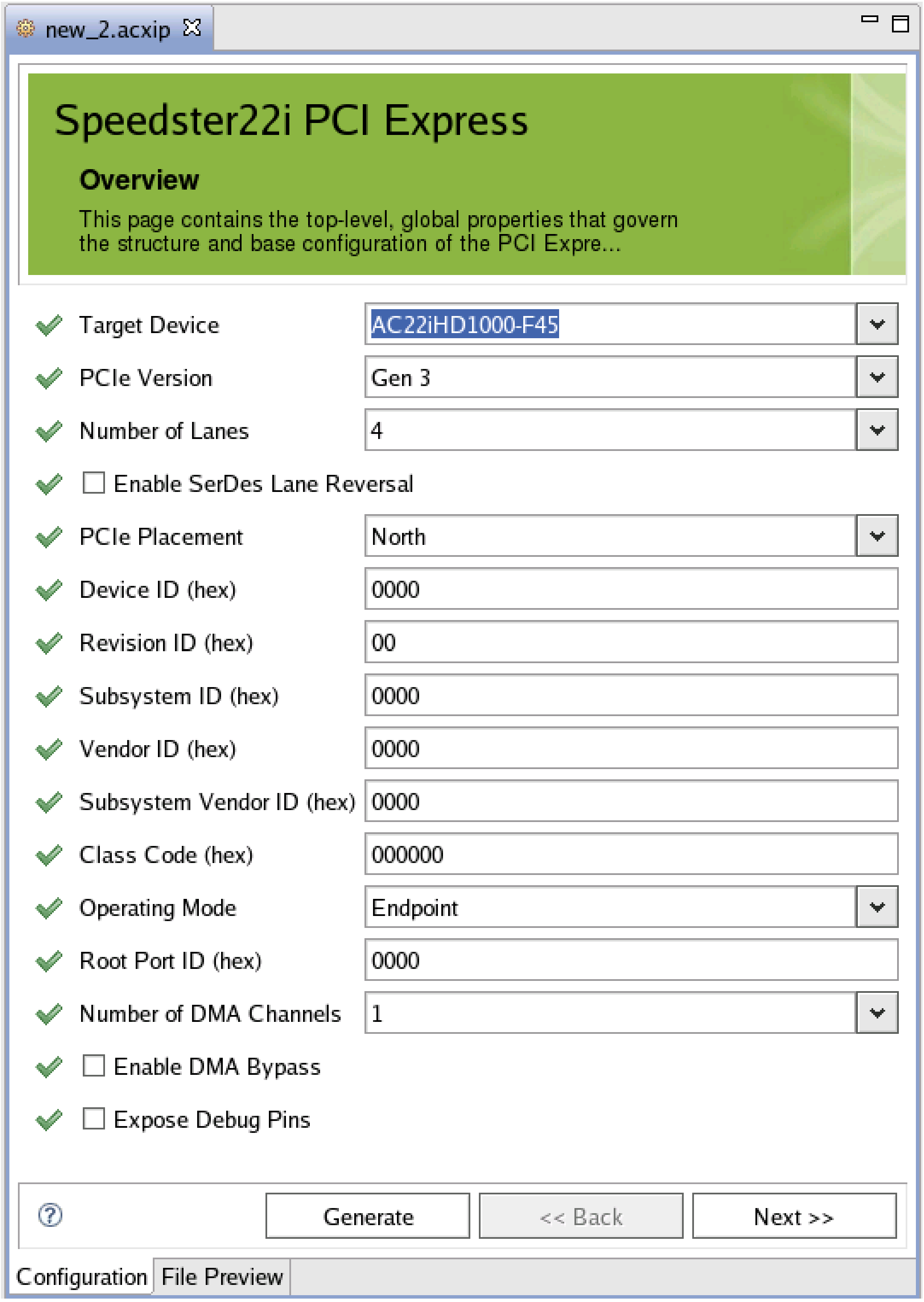


Table 33: *PCIe Editor Overview Page Options*

Option	Description
Target Device	The Target Device allows you to select from any compatible Speedster22i devices for placing the PCI Express core and SerDes lanes.
PCIe Version	You can choose which PCI Express standard you want to use: Gen 1, Gen 2, or Gen 3
Number of Lanes	The number of serdes lanes to use
Enable SerDes Lane Reversal	Reverse the SerDes lanes that get mapped to lanes 0-7 as lanes 7 - 0
PCIe Placement	Choose which site this PCIe instance should occupy. The Placement IP Diagram will be updated to show the chosen configuration.
Device ID (hex)	The PCI Express Device ID
Revision ID (hex)	The PCI Express Revision ID
Subsystem ID (hex)	The PCI Express Sub-system ID
Vendor ID (hex)	The PCI Express Vendor ID
Subsystem Vendor ID (hex)	The PCI Express Subsystem Vendor ID
Class Code (hex)	Value returned when the Class Code Configuration Register is read. Must be set to the correct value for the type of device being implemented
Operating Mode	The operating mode of the PCI Express traffic
Root Port ID (hex)	This 16 bit field is used to define the ID used for PCIe Requester ID and Completer ID when the core is operating as a Downstream Port (Root Port, Downstream Switch Port). When the core is operating as an Upstream Port (Endpoint, Upstream Switch Port), the core captures its Requestor/Completer ID from received Configuration Write transactions.

Option	Description
Number of DMA Channels	The number of DMA channels for this interface.
Enable DMA Bypass	Bypass the DMA interface and use only the bypass interface.

Memory Map Page

This page contains the options that pertain to the PCIe Memory Map.

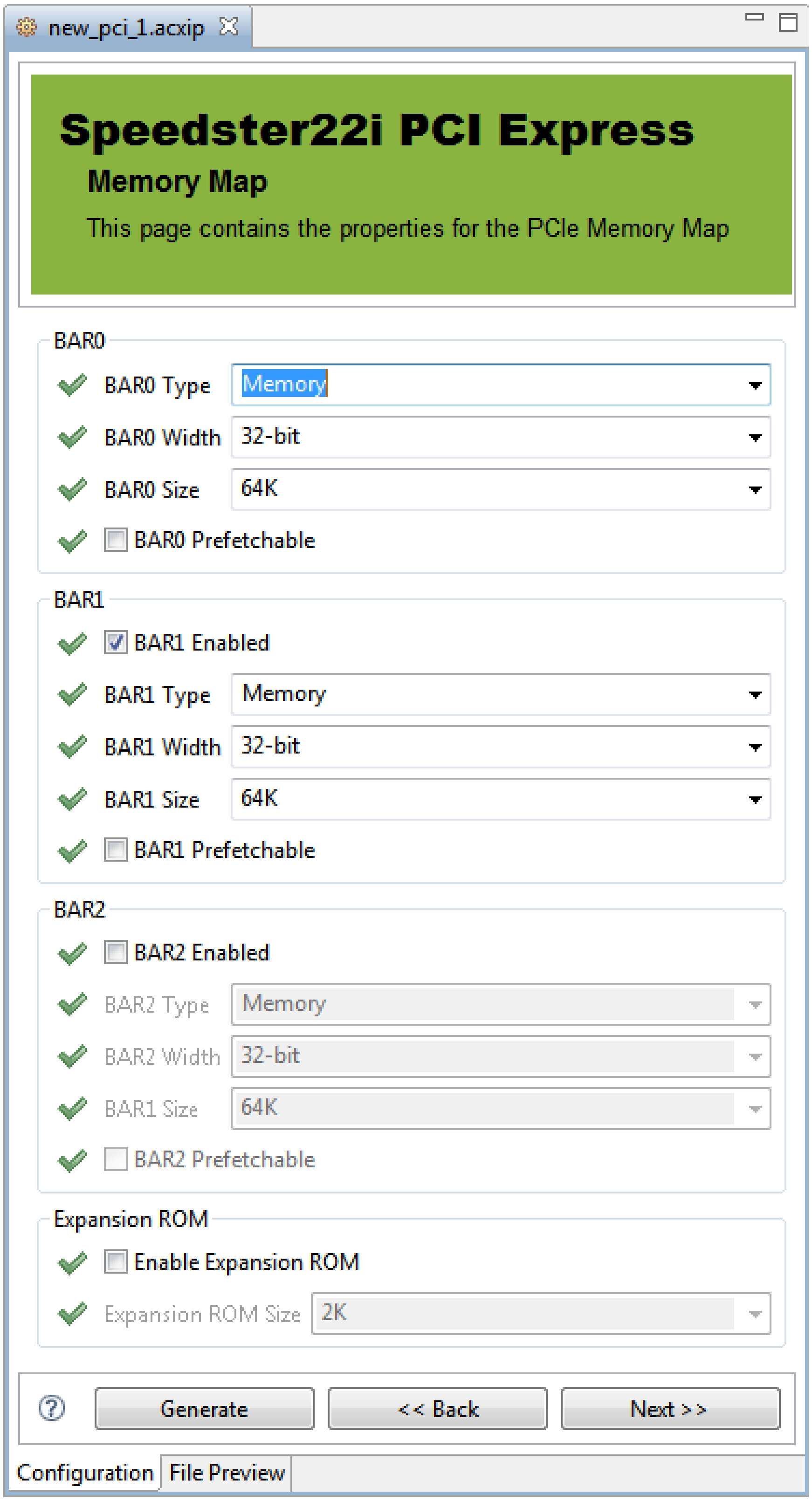

new_pci_1.acxip
Speedster22i PCI Express
Memory Map
This page contains the properties for the PCIe Memory Map
BAR0
BAR0 Type
Memory
BAR0 Width
32-bit
BAR0 Size
64K
BAR0 Prefetchable
BAR1
BAR1 Enabled
BAR1 Type
Memory
BAR1 Width
32-bit
BAR1 Size
64K
BAR1 Prefetchable
BAR2
BAR2 Enabled
BAR2 Type
Memory
BAR2 Width
32-bit
BAR1 Size
64K
BAR2 Prefetchable
Expansion ROM
Enable Expansion ROM
Expansion ROM Size
2K
Generate
<< Back
Next >>
Configuration
File Preview

Table 34: *PCIe Editor Overview Page Options*

Option	Description
BAR0	
BAR0 Type	Each BAR can be configured as Memory or I/O
BAR0 Width	Each BAR can be configured as 32-bit or 64-bit
BAR0 Size	The size of the BAR in bytes. A minimum of 4K bytes is recommended
BAR0 Prefetchable	This enables/disables prefetch of the BAR, and should only be enabled for 64-bit BARs
BAR1	
BAR1 Enabled	Enable use of BAR1
BAR1 Type	Each BAR can be configured as Memory or I/O
BAR1 Width	Each BAR can be configured as 32-bit or 64-bit
BAR1 Size	The size of the BAR in bytes. A minimum of 4K bytes is recommended
BAR1 Prefetchable	This enables/disables prefetch of the BAR, and should only be enabled for 64-bit BARs
BAR2	
BAR2 Enabled	Enable use of BAR2
BAR2 Type	Each BAR can be configured as Memory or I/O
BAR2 Width	Each BAR can be configured as 32-bit or 64-bit
BAR2 Size	The size of the BAR in bytes. A minimum of 4K bytes is recommended
BAR2 Prefetchable	This enables/disables prefetch of the BAR, and should only be enabled for 64-bit BARs
Expansion ROM	
Enable Expansion ROM	The Expansion ROM BAR is used to store device specific initialization or boot instructions that must execute during the boot process. Use of the Expansion ROM Base Address is rare. If implemented a valid Expansion ROM structure must be implemented at this BAR location or the system may fail to boot.

Option	Description
Expansion ROM Size	The size of the expansion ROM in bytes.

Power Management Page

This page contains all the options for PCIe power management.

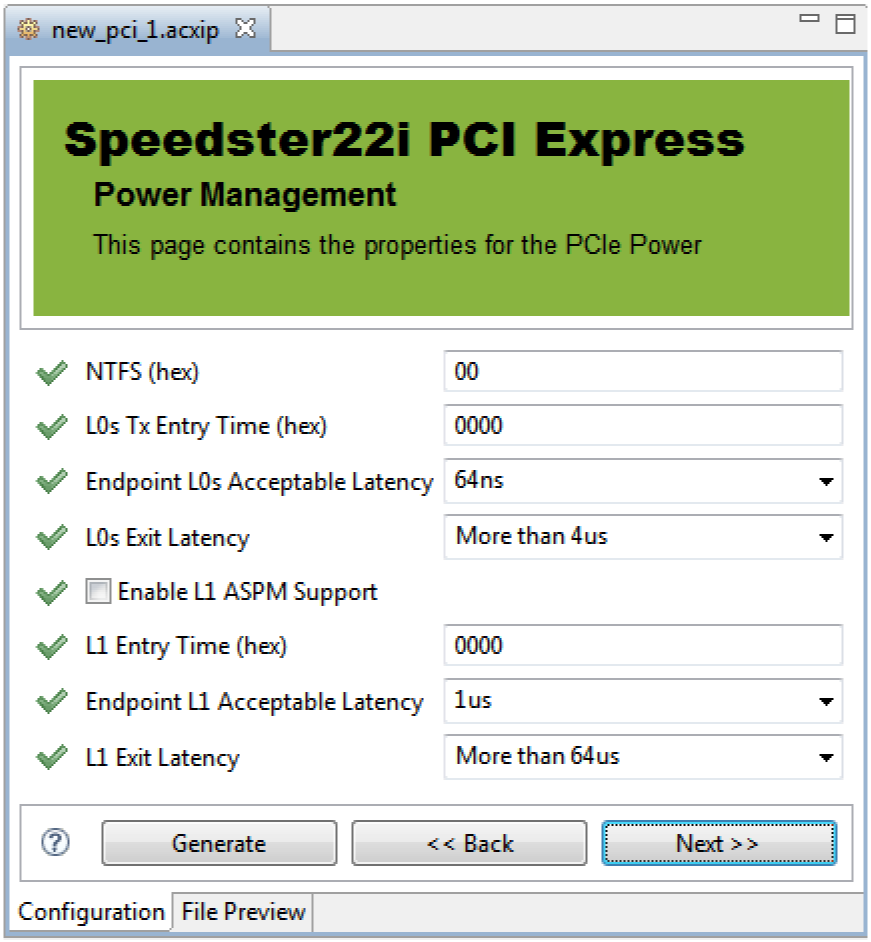


Table 35: ***PCIe Editor Power Management Page Options***

Option	Description
NTFS (hex)	Number of NFTS sets to request when exiting L0s. This is the NFTS value transmitted in TS1 and TS2 Ordered Sets during training.
L0s Tx Entry Time (hex)	Number of nanoseconds of idle time to wait before entering L0s TX. Idle time is defined as no TLP or DLLP transmission pending or actively being transmitted. By PCIe Specification, the value programmed should be ≤ 7uS (0x1B58). Too low a value risks wasting link bandwidth due to L0s entry/exit latencies. Too high a value will reduce L0s power savings.

Option	Description
Endpoint L0s Acceptable Latency	From PCI Express Base Specification, Rev 2.1 section 7.8.3: Acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering.
L0s Exit Latency	Length of time required to complete transition from L0s to L0
Enable L1 ASPM Support	Active State Power Management (ASPM) Support
L1 Entry Time (hex)	Number of microseconds of idle time to wait before requesting entry to ASPM L1 (used by Upstream Ports â€“ Endpoint/Upstream Switch - only). Idle time is defined as no TLP or ACK/NAK DLLP transmissions. PCIe Specification does not define a minimum or maximum value. Too low a value risks wasting link bandwidth due to ASPM L1 entry/exit latencies. Too high a value will reduce ASPM L1 power savings. Only used if Enable L1s Power Mgmt is set. 0 is a special case and selects 1000 uS (0x3e8)
Endpoint L1 Acceptable Latency	From PCI Express Base Specification, Rev 2.1 section 7.8.3: This field indicates the acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering
L1 Exit Latency	Length of time required to complete transition from L1 to L0

Advanced Features Page

This page contains all the options that govern the advanced features of the PCI Express interface.

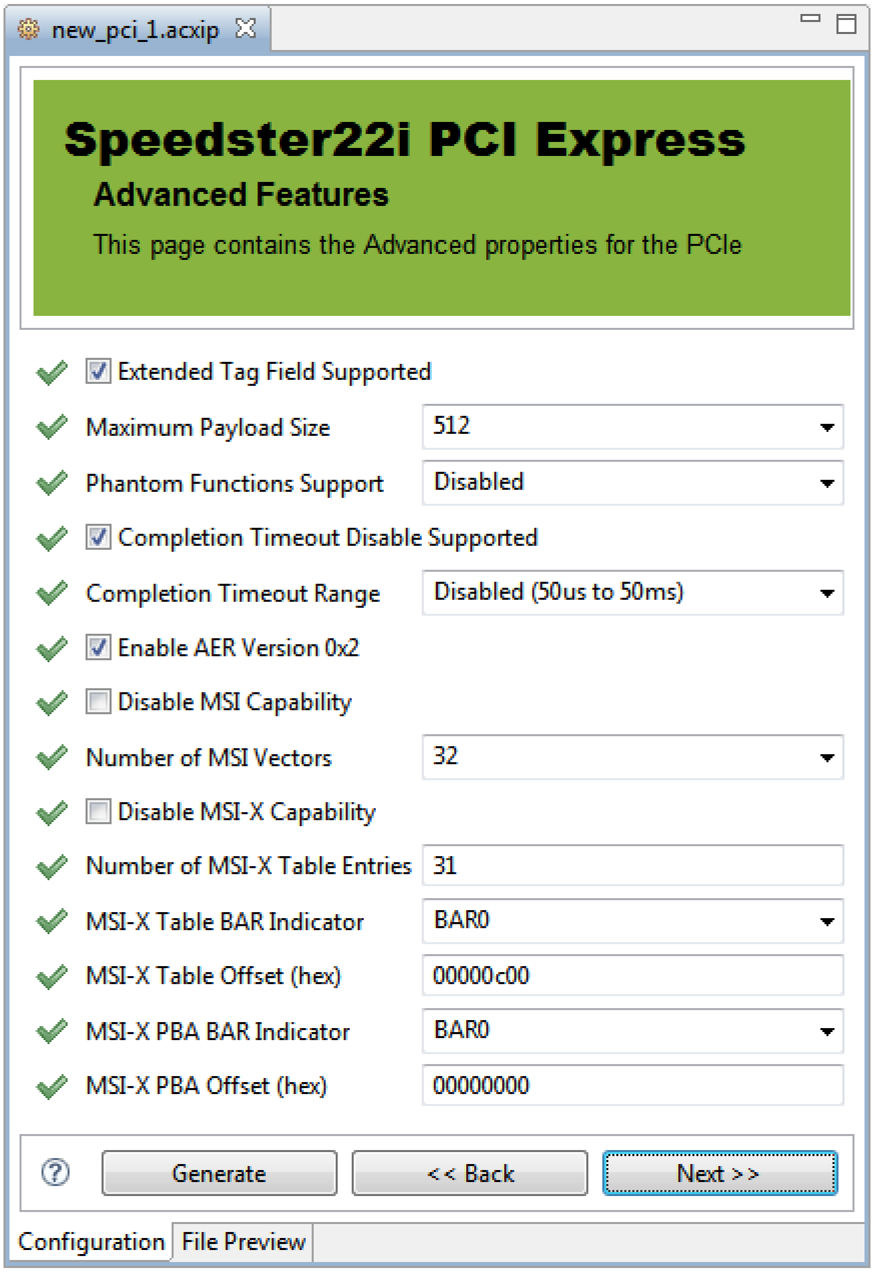


Table 36: *PCIe Editor Advanced Features Page Options*

Option	Description
Extended Tag Field Supported	Enable Extended Tag Field Support
Maximum Payload Size	Sets the maximum payload size supported

Option	Description
Phantom Functions Support	Phantom Function support for the Function must be enabled by the Phantom Functions Enable field in the Device Control register before the Function is permitted to use the Function Number field in the Requester ID for Phantom Functions. If Phantom Functions Supported != 00, the core implements the Phantom Functions Enable register as read/write resetting to 0 and otherwise implements Phantom Functions Enable as read only tied to 0.
Completion Timeout Disable Supported	Set to signal that user Completion Timeout mechanism supports being disabled; clear to indicate that the user Completion Timeout mechanism may not be disabled. Setting this bit is required by PCIe Spec. for Endpoints which issue requests on their own behalf so 1 is the recommended value.
Completion Timeout Range	The supported completion timeout range. Devices are not required to support several timeout ranges. 50uS to 50mS is the recommended value.
Enable AER Version 0x2	1 == Implement AER to version 0x2 (PCIe 2.1 and later Specification revisions). Correctable Errors: Corrected Internal Error & Header Log Overflow are enabled. Uncorrectable Error: Uncorrected Internal Error is enabled. 0 == Implement AER to version 0x1 (PCIe 2.0 and earlier Specification revisions). Correctable Errors: Corrected Internal Error & Header Log Overflow are hidden and cannot be signaled. Uncorrectable Error: Uncorrected Internal Error is hidden and cannot be signaled.
Disable MSI Capability	When disabled, the core's MSI Capability is removed from the Configuration Registers Capabilities List, MSI Interrupt functionality is disabled, and it will not be possible to send MSI interrupts
Number of MSI Vectors	Multiple message MSI functionality requires the user design to indicate the interrupt vector number that they want signaled when mgmt_interrupt is asserted. MSI Multiple Message Capable advertises the desired number of vectors. System software is not required to provide the desired number of vectors and programs the allocated number of vectors into the Multiple Message Enable configuration register.
Disable MSI-X Capability	When disabled, the core's MSI-X Capability is removed from the Configuration Registers Capabilities List, MSI-X Interrupt functionality is disabled, and it will not be possible to send MSI-X interrupts; this bit only affects core configurations that support MSI-X
Number of MSI-X Table Entries	MSI-X functionality requires the user design to implement the MSI-X Table in Memory Space. MSI-X Table Size [10:0] is set to indicate the number of MSI-X Table entries (Interrupt Vectors) implemented. MSI-X Table Size is read by software to determine the size of the MSI-X Table.
MSI-X Table BAR Indicator	MSI-X functionality requires the user design to implement the MSI-X Table in Memory Space mapped by 1 (32-bit) or 2 (64-bit) Memory Base Address Registers. MSI-X Table BIR and MSI-X Table Offset indicate to system software where the MSI-X Table is located.
MSI-X Table Offset (hex)	Value to place into MSI-X Capability : Table Offset field. MSI-X Table BIR indicates which Base Address Register contains the MSI-X Table

Option	Description
MSI-X PBA BAR Indicator	Same as MSI-X Table BIR, but indicates the Base Address Register of the MSI-X PBA rather than the MSI-X Table
MSI-X PBA Offset (hex)	Same as MSI-X Table Offset, but indicates the Base Address Register offset for the MSI-X PBA rather than the MSI-X Table

Gen 3 Equalization Page

This page contains all the options that govern Gen 3 PCI Express equalization.

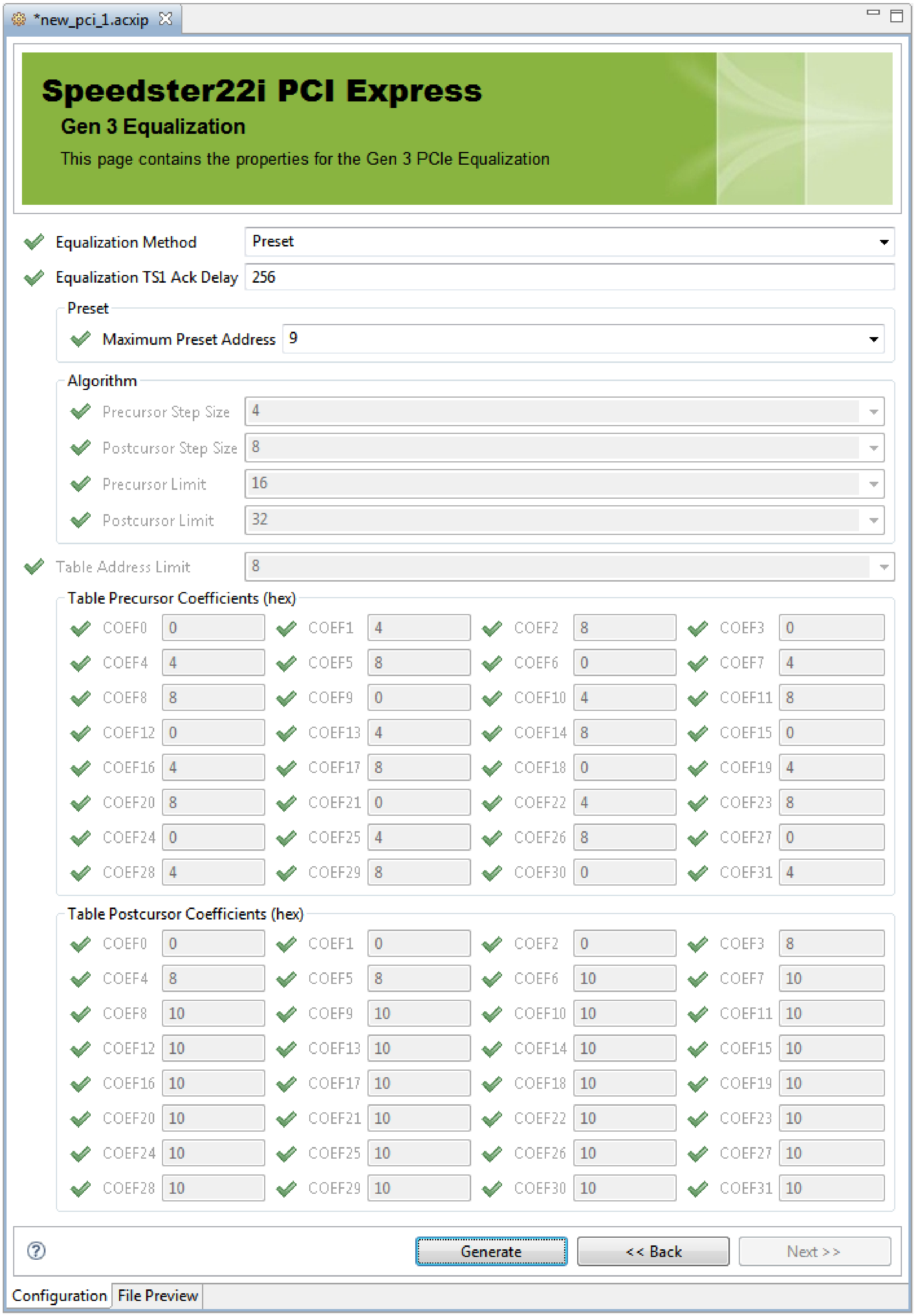

*new_pci_1.acxip
Speedster22i PCI Express
Gen 3 Equalization
This page contains the properties for the Gen 3 PCIe Equalization
Equalization Method Preset
Equalization TS1 Ack Delay 256
Preset
Maximum Preset Address 9
Algorithm
Precursor Step Size 4
Postcursor Step Size 8
Precursor Limit 16
Postcursor Limit 32
Table Address Limit 8
Table Precursor Coefficients (hex)
COEF0 0 COEF1 4 COEF2 8 COEF3 0
COEF4 4 COEF5 8 COEF6 0 COEF7 4
COEF8 8 COEF9 0 COEF10 4 COEF11 8
COEF12 0 COEF13 4 COEF14 8 COEF15 0
COEF16 4 COEF17 8 COEF18 0 COEF19 4
COEF20 8 COEF21 0 COEF22 4 COEF23 8
COEF24 0 COEF25 4 COEF26 8 COEF27 0
COEF28 4 COEF29 8 COEF30 0 COEF31 4
Table Postcursor Coefficients (hex)
COEF0 0 COEF1 0 COEF2 0 COEF3 8
COEF4 8 COEF5 8 COEF6 10 COEF7 10
COEF8 10 COEF9 10 COEF10 10 COEF11 10
COEF12 10 COEF13 10 COEF14 10 COEF15 10
COEF16 10 COEF17 10 COEF18 10 COEF19 10
COEF20 10 COEF21 10 COEF22 10 COEF23 10
COEF24 10 COEF25 10 COEF26 10 COEF27 10
COEF28 10 COEF29 10 COEF30 10 COEF31 10
Generate
<< Back
Next >>
Configuration
File Preview

Table 37: *PCIe Editor Gen 3 Equalization Page Options*

Option	Description
Equalization Method	The Equalization method to use: Preset, Algorithm, or Table
Equalization TS1 Ack Delay	Defines how long the upstream port (Phase 2) or downstream port (Phase 3) waits after requesting new coefficients/presets before looking for incoming EQ TS1 sets from the remote link partner. This delay by specification should be set to the round trip delay to the remote link partner (including logic delays in the requesting port) + 500ns.
Preset	
Maximum Preset Address	Step through the PCI Express Specification-defined Tx Presets (0 through 9). The Preset method trying all presets 0 to 9 is recommended for users to start with if they are unsure which method they should use.
Algorithm	
Precursor Step Size	The algorithm precursor step size
Postcursor Step Size	The algorithm postcursor step size
Precursor Limit	The algorithm precursor limit
Postcursor Limit	The algorithm postcursor limit
Table Address Limit	The table address limit. Be careful when assigning the table address limit not to exceed the Equalization time limit.
Table Precursor Coefficients (hex)	
COEF0	
COEF1	
…	…
COEF31	
Table Postcursor Coefficients (hex)	
COEF0	

Option	Description
COEF1	
…	…
COEF31	

Speedster22i PIPE Configuration Editor

The PIPE Configuration Editor provides a simplified graphical wizard for creating a PIPE wrapper IP configuration file (. acxip) for the SerDes interface. This editor allows the user to generate the required configuration files for design with the embedded PIPEs. See Creating an IP Configuration (see page 387).

Once the user has configured the PIPE wrapper to meet their requirements, and the PIPE Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

By default, the PIPE Configuration Editor is included in the IP Configuration perspective (**Window** → **Open Perspective** → **IP Configuration**).

IP Diagrams

The PIPE Configuration editor includes both a module interface diagram, and a placement diagram.

The IP Diagram View (see page 196)'s Placement Diagram shows the currently-selected placement of the PIPE module according to the settings selected in the Editor. The user is able to click on potential PIPE placement locations in this diagram, and both the Editor and Diagram will be updated to show the new placment, assuming the user clicked in a valid location. Note that the PIPE requires all consumed lanes to be contiguous.

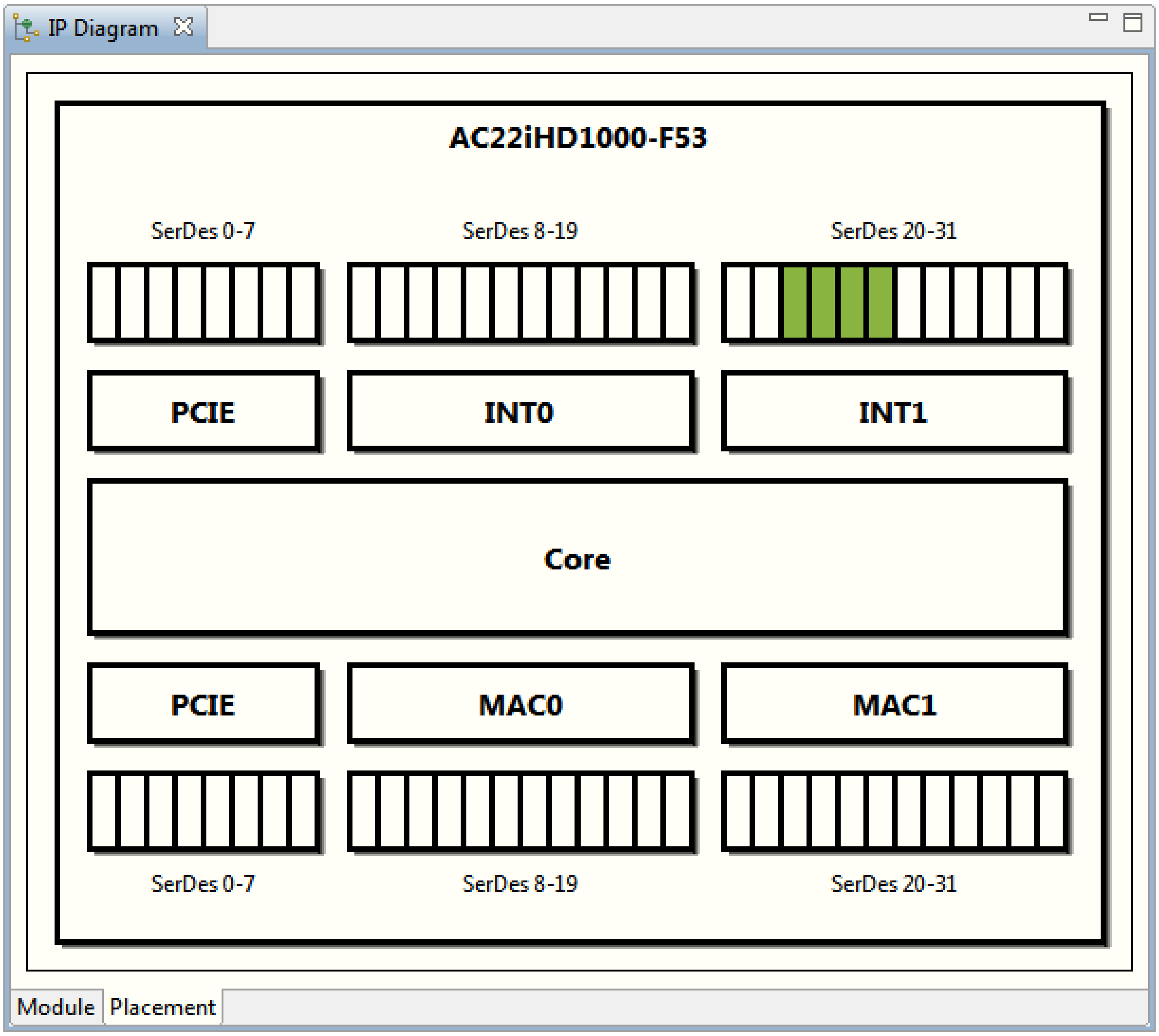


Figure 7: PIPE Placement Diagram (four lanes, North 22-25)

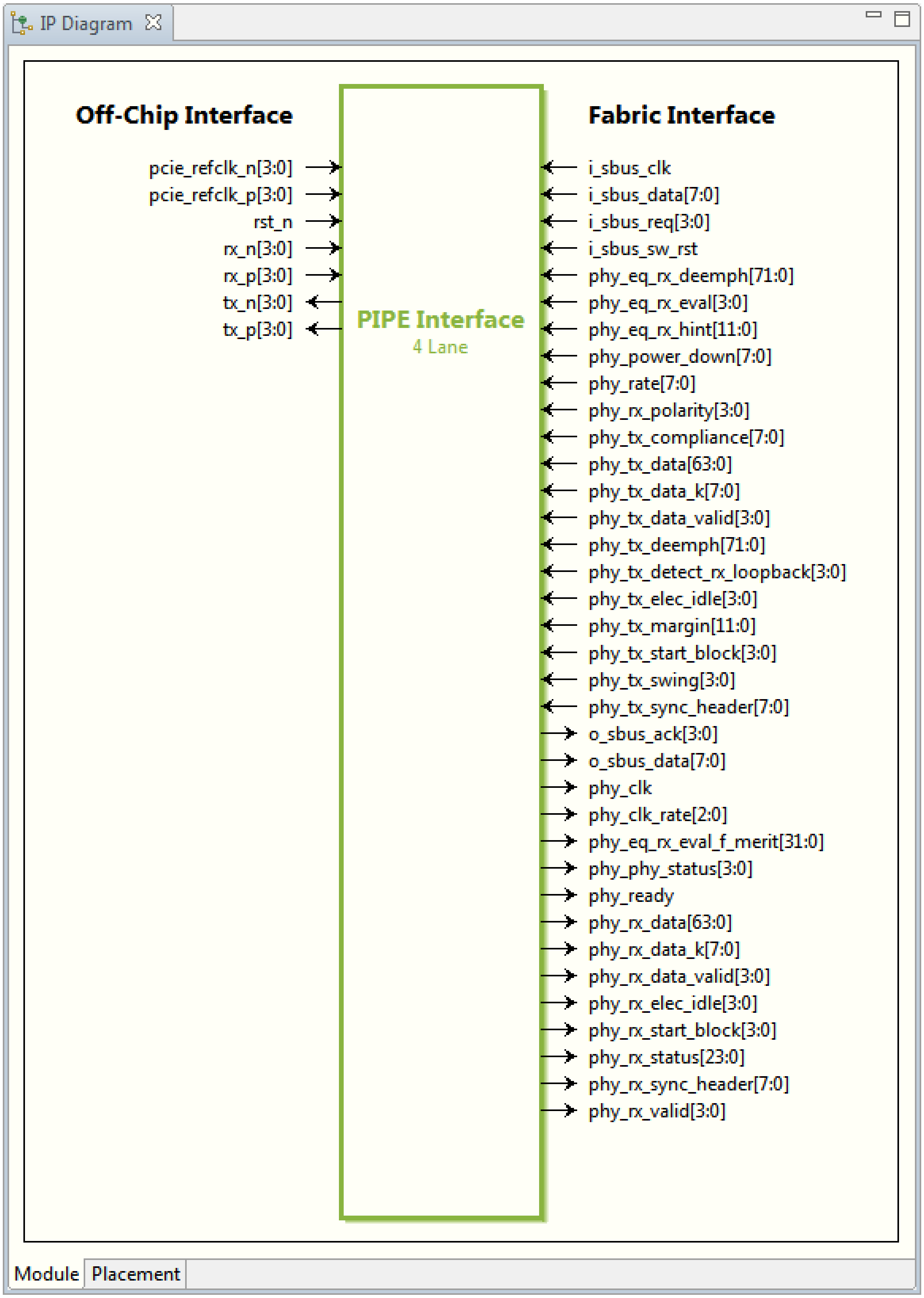


Figure 8: PIPE Module Diagram (four lane)

The IP Diagram View's Module Diagram for the PIPE shows live information about the current configuration in the Editor, including the total lane count and which inputs and outputs are currently active. Additionally, relevant configuration errors will be shown with a red background, and configuration warnings will be shown with a yellow background (these are the default IP Diagram colors, and may be modified in the Preferences through the IP Diagram Preference Page (see page 300)).

PIPE Overview Page

The Overview page of the Speedster22i PIPE Configuration Editor (see page 113) contains the top-level, global properties that govern the structure and base configuration of the PIPE.

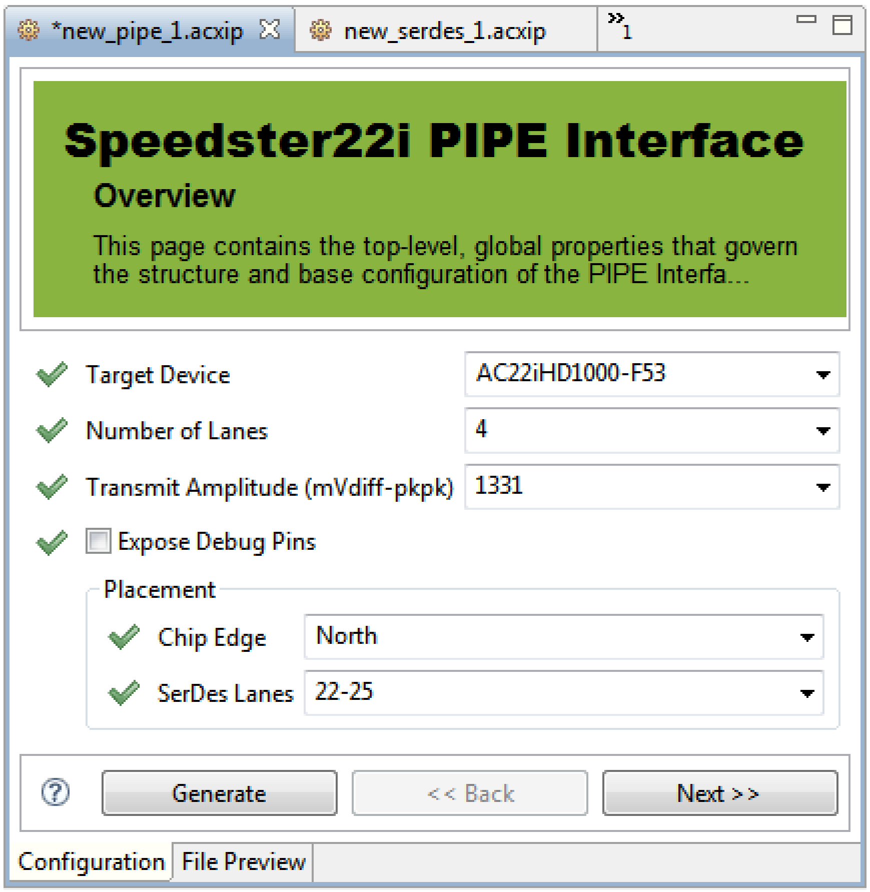


Figure 9: PIPE Overview Page

Option	Description
Target Device	The Target Device allows you to select from any compatible Speedster22i devices for placing the PIPE Interface SerDes lanes.
Number of Lanes	Selects the number of SerDes lanes for this interface.
Transmit Amplitude (mVdiff-pkpk)	Defines the full-scale maximum swing of the PMA Tx driver
Expose Debug Pins	Expose debug output pins from the SerDes

Option	Description
Placement	
Chip Edge	This property selects whether the SerDes interface is placed along the North or South edge of the chip
SerDes Lanes	This property selects which lanes the SerDes interface is placed on

Ref Clock Termination Page

The Ref Clock Termination page of the Speedster22i PIPE Configuration Editor (see page 113) allows the user to configure the SerDes Ref Clock Termination values on a per-lane basis. The number of options will vary according to the number of lanes used by the PIPE interface.

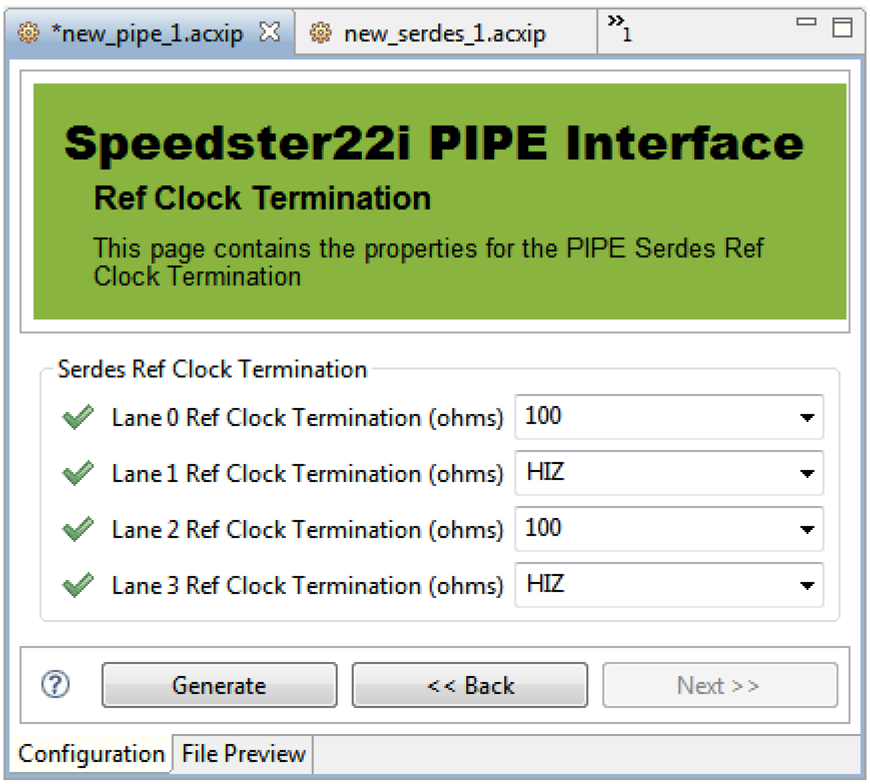


Figure 10: PIPE Ref Clock Termination Page (four lanes)

Option	Description
SerDes Ref Clock Termination	
Lane *N* Ref Clock Termination (ohms)	SerDes ref clock termination impedance setting, in Ohms. (There will be an option for each of the utilized lanes.)

Speedster22i ROM Configuration Editor

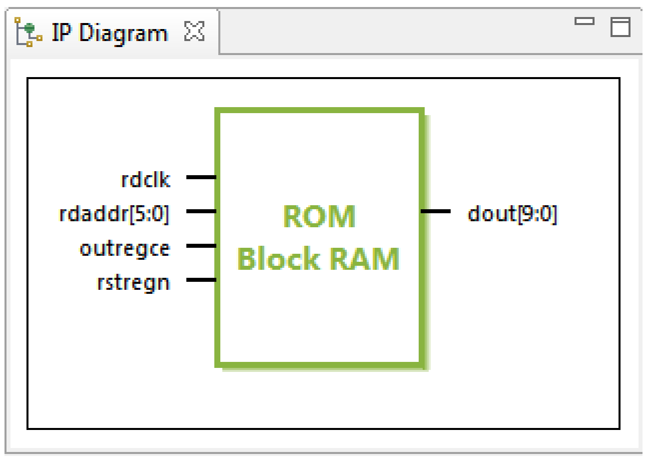
The ROM configuration editor provides a simple graphical editor used to configure a ROM wrapper instance, and saves the user configuration in a ROM IP configuration file (.acxip). See Creating an IP Configuration (see page 387).

Once the user has configured the IP to meet their requirements, and the Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

By default, the ROM Configuration Editor is included in the  IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**).

IP Diagram

The IP Diagram View (see page 196) for the ROM config editor will display a module diagram, showing all the inputs and outputs of the ROM instance according to the current editor configuration.



ROM Overview Page

The Overview page contains all the properties that govern the structure and configuration of the ROM wrapper.

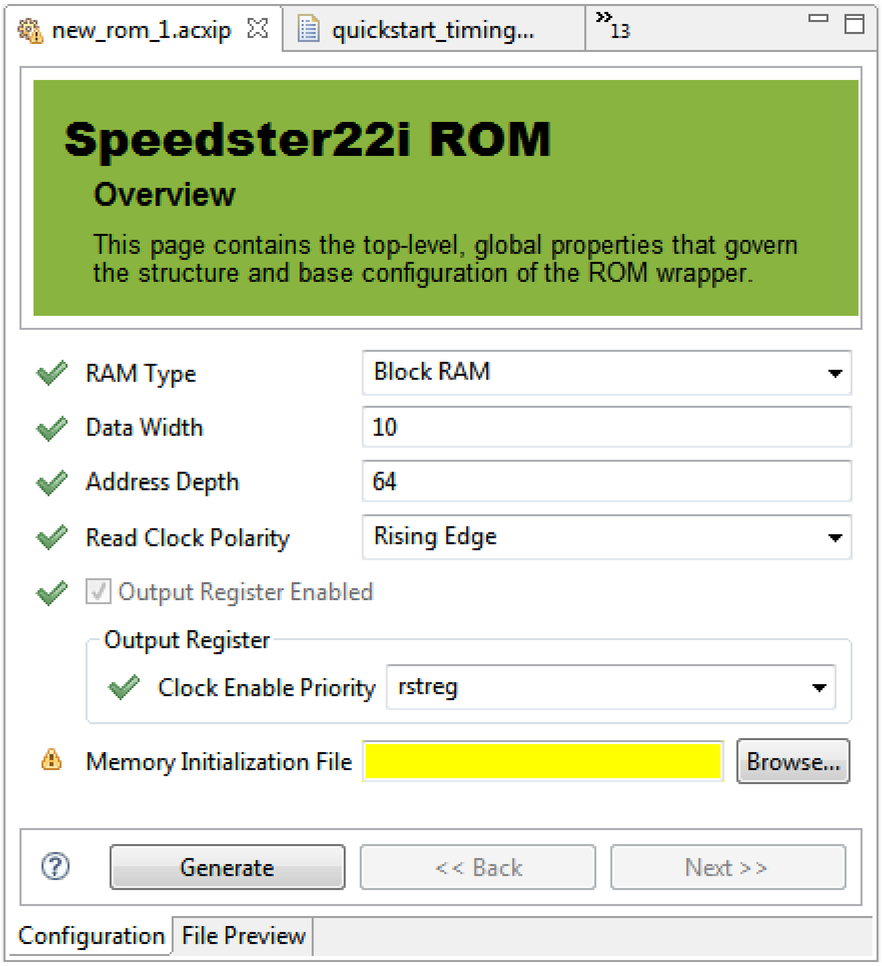


Table 38: *ROM Editor Overview Page Options*

Option	Description
RAM Type	The ROM can be built out of Block RAMs or Local RAMs.
Data Width	Data width of the read port.
Address Depth	Desired address depth of the ROM.
Read Clock Polarity	The read port clock polarity can be set to use either rising edge assignment or falling edge assignment.
Output Register Enabled	When the Output Register is enabled, there is an additional cycle of latency for each read operation.
Output Register	

Option	Description
Clock Enable Priority	The Clock Enable Priority defines the priority of the `outregce` clock enable input relative to the `rstreg` reset input during an assertion of the `rstreg` signal on the read port output register. Setting this field to **rstreg** allows the output register to be set/reset at the next active edge of the read port clock without requiring a specific value on the `outregce` output register clock enable input. Setting this field to **regce** requires that the `outregce` output register clock enable input is high for the output register set/reset operation to occur at the next active edge of the read port clock.
Memory Initialization File	Path to initialization file whose data is “Data Width” wide and “Address Depth” deep. The memory initialization file should be in hexadecimal.

Speedster22i SerDes Configuration Editor

The SerDes Configuration Editor provides a graphical wizard for creating a SerDes IP configuration file (`.acxip`). This view allows the user to generate the required configuration files for design with the embedded 12G SerDes. Pages are accessed via **«Back** and **Next»** buttons.

By default, the SerDes Configuration Editor is included in the IP Configuration Perspective (**Window -> Open Perspective -> IP Configuration**).

See also: Creating an IP Configuration (see page 387)

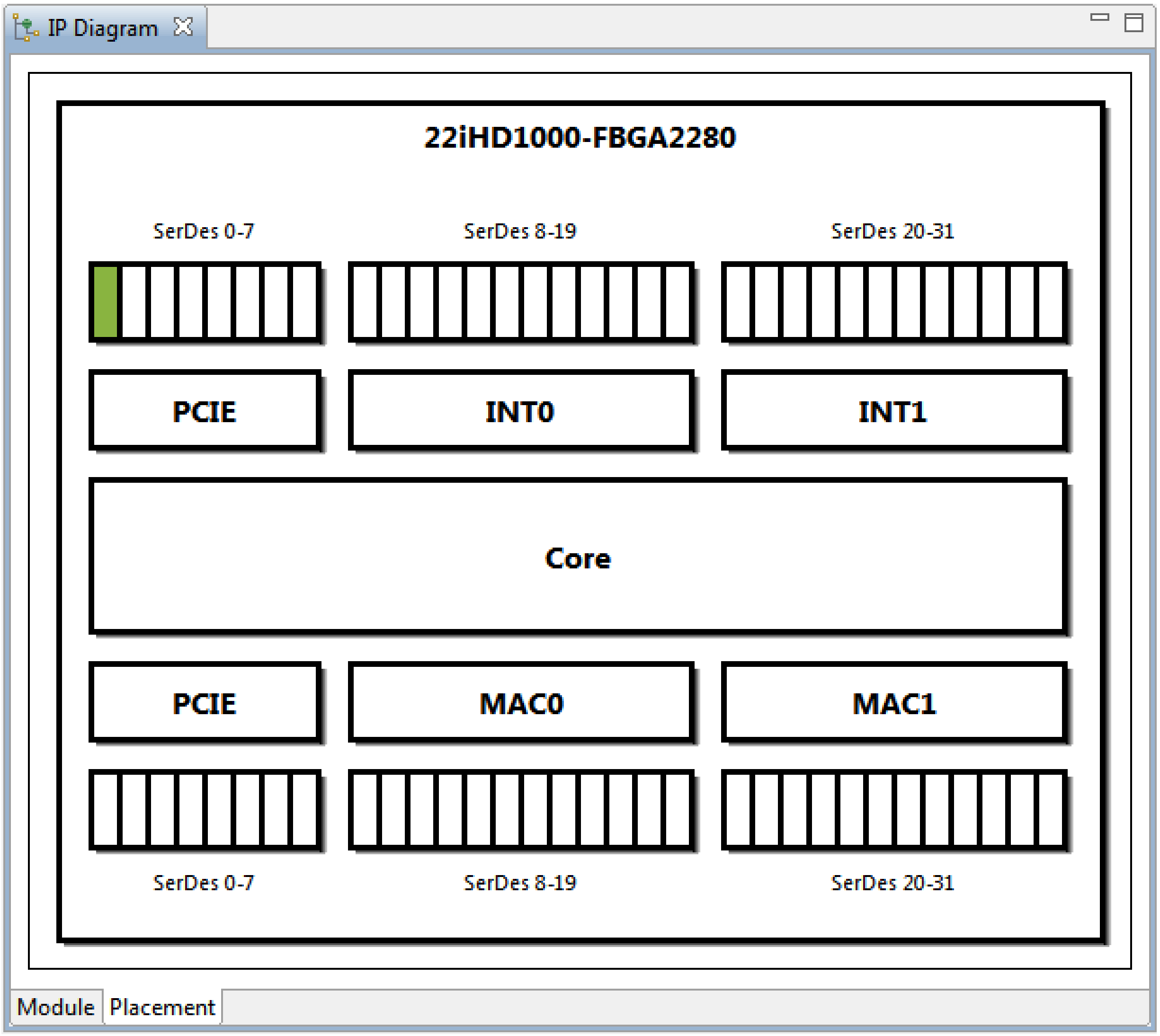
IP Diagrams

In the IP Diagram view for the SerDes Configuration Editor, there are two tabs (at the bottom) allowing the user to view the two types of IP diagrams for the SerDes: Placement and Module.

Placement Diagram

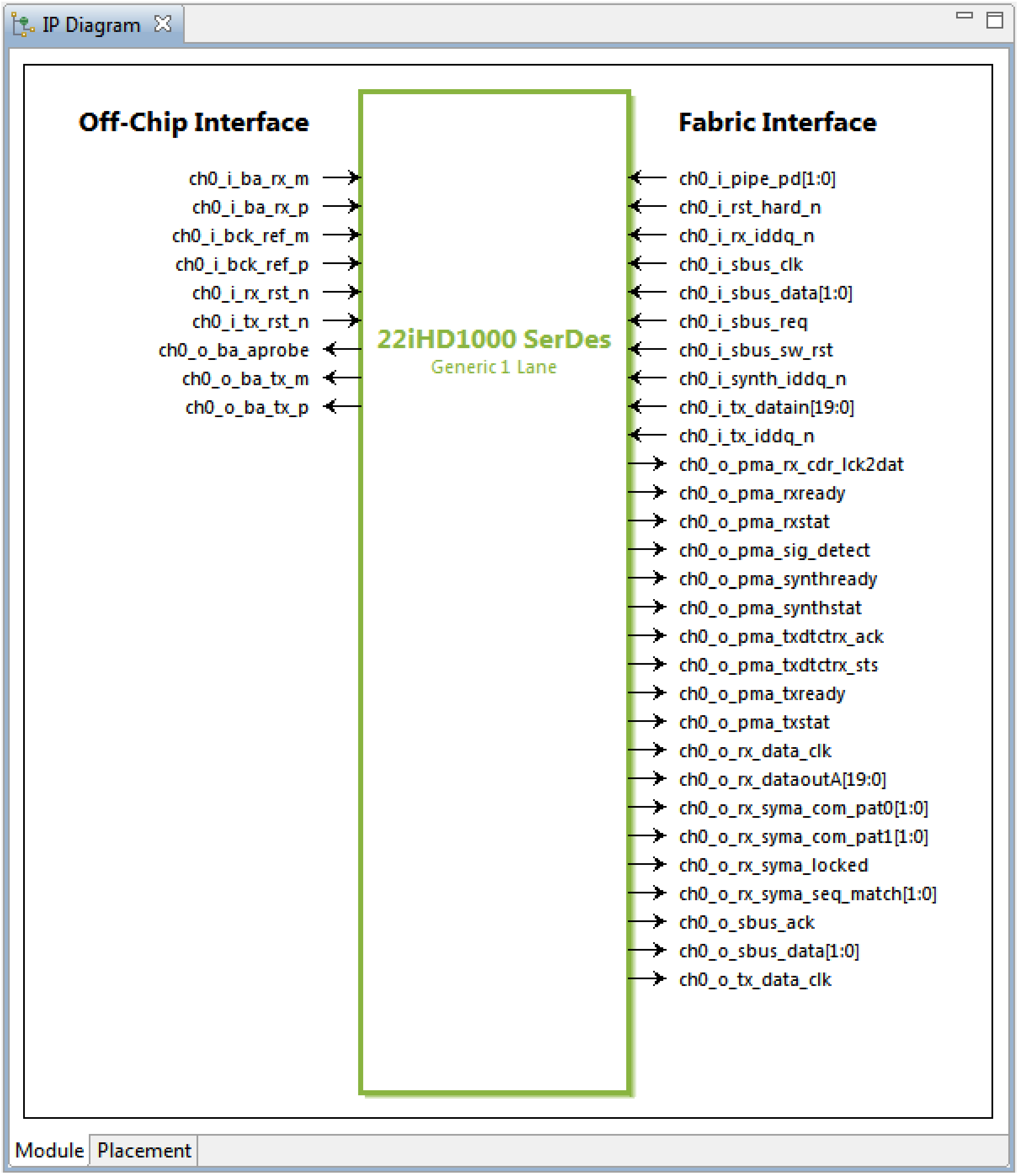
The IP Diagram View (see page 196)'s Placement Diagram shows the placement of the SerDes as currently selected in the Editor. It also allows the user to click on the placement they want in the diagram itself, and the placement settings in the Editor will be updated accordingly.

> Lanes which are grey (disabled) in the placement diagram are not available in the current package / configuration.



Module Diagram

The Module Diagram shows the inputs and outputs of the SerDes instance currently being configured in the Editor.



SerDes Overview Page

This Overview page contains the top-level, global properties that govern the structure and base configuration of the 12G SerDes wrapper.

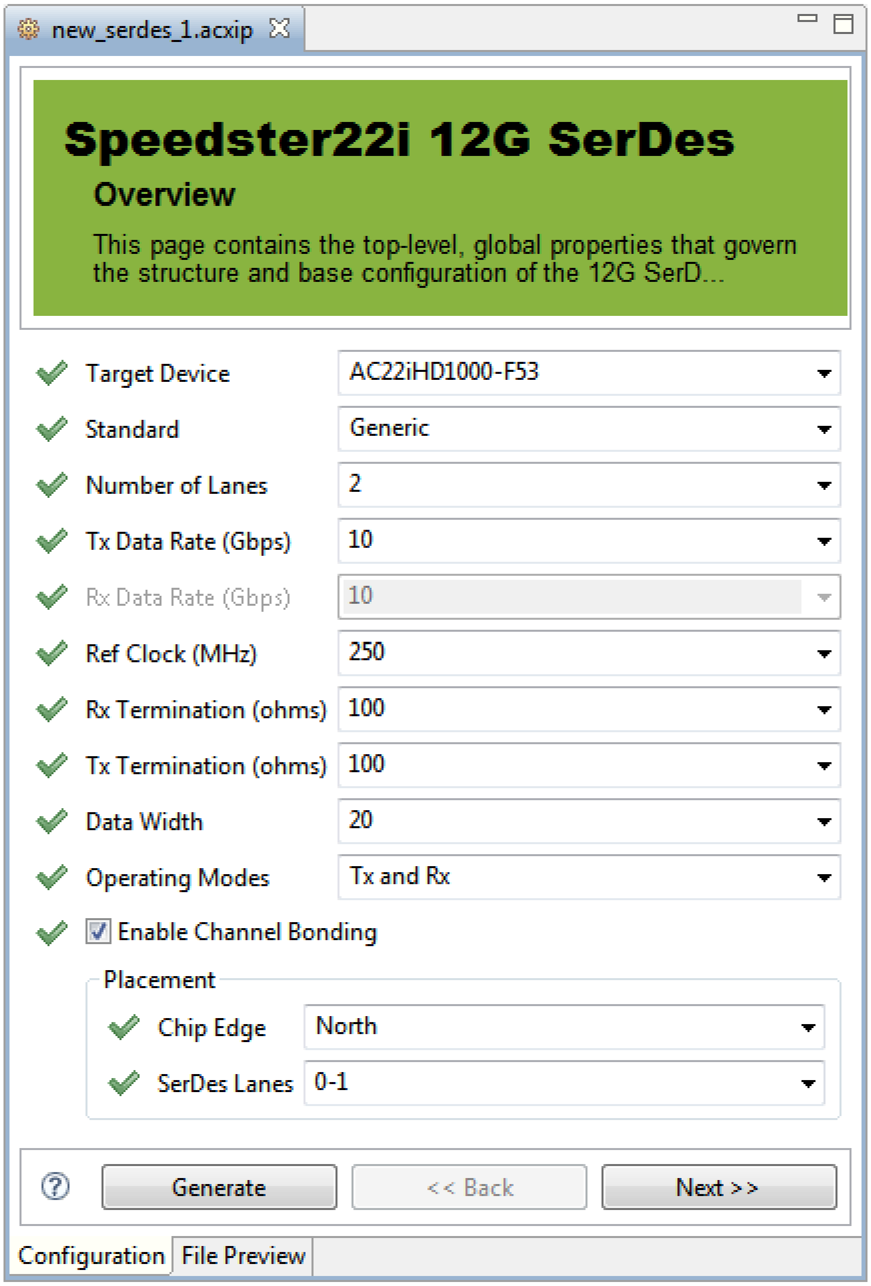


Table 39: *Overview Page Options*

Option	Description
Target Device	The Target Device allows you to select from any compatible Speedster22i devices for placing SerDes Lanes.
	The Standard property sets the overall SerDes standard used in this configuration. Setting the Standard to a pre-defined (non-generic) standard will automatically configure other SerDes properties to comply with the pre-defined standard. Setting the standard to **Generic** will allow the user to create a fully customized configuration.

Option		Description
Standard		Changing this value will cause many other SerDes options on this configuration page and other configuration pages to immediately be altered to default values which are compatible with the chosen standard. Options which are completely incompatible with the standard will be disabled. Other options may change their range of allowed values. If the user has altered options in this configuration prior to changing the Standard, the user must be aware that prior changes will likely be lost.
Number of Lanes		Sets the number of lanes to be used.
Tx Data Rate (Gbps)		Desired transmit-side data rate.
Rx Data Rate (Gbps)		Sets the receive-side data rate.
Ref Clock (MHz)		Sets the reference clock frequency to be delivered to the SerDes.
Rx Termination (ohms)		Rx Lane calibration impedance setting
Tx Termination (ohms)		Tx Lane calibration impedance setting
Data Width		Data bus interface width to the SerDes.
Operating Modes		Defines whether transmit function, receive function, or both are enabled.
Enable Channel Bonding		Enables channel bonding and deskew control between lanes.
Placement		
	Chip Edge	Selects whether the SerDes interface is placed along the North or South edge of the chip.
	SerDes Lanes	The lanes in which the SerDes interface is placed.

See also: Creating an IP Configuration (see page 387).

PMA Settings Page

Allows the user to select a uniform or lane-based PMA.

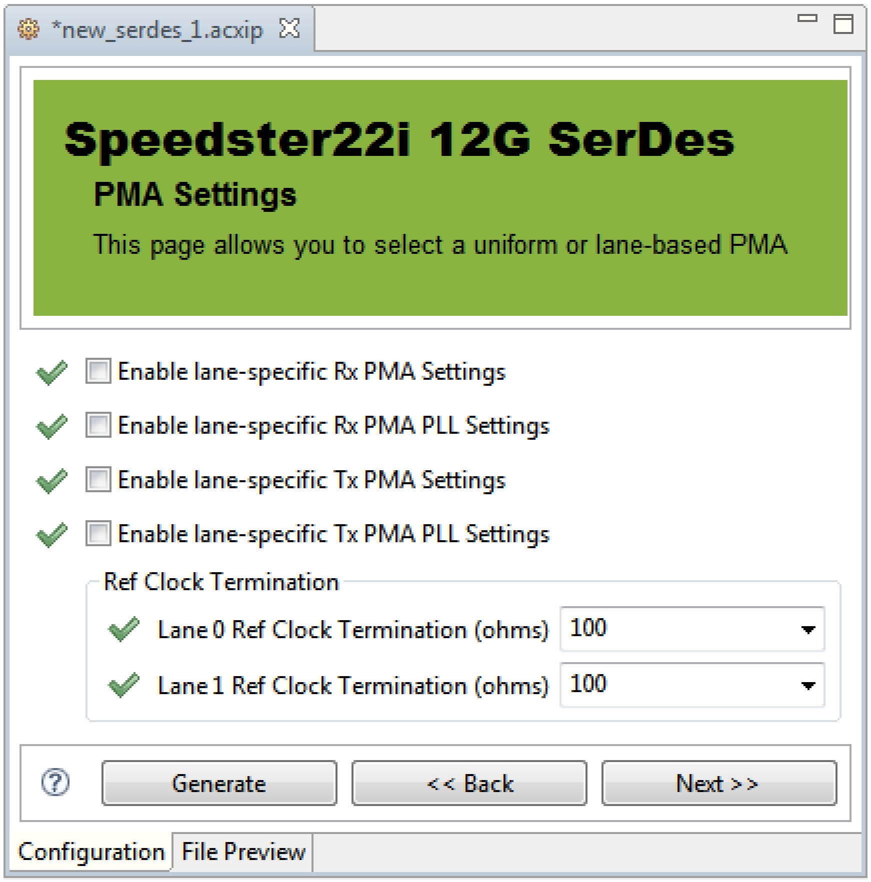


Table 40: ***PMA Settings Page Options***

Option	Description
Enable lane-specific Rx PMA Settings	If enabled, instead of a single page configuring the Rx PMA settings for all lanes, each lane will have its own page.
Enable lane-specific Rx PMA PLL Settings	If enabled, instead of a single page configuring the Rx PMA PLL settings for all lanes, each lane will have its own page.
Enable lane-specific Tx PMA Settings	If enabled, instead of a single page configuring the Tx PMA settings for all lanes, each lane will have its own page.
Enable lane-specific Tx PMA PLL Settings	If enabled, instead of a single page configuring the Tx PMA PLL settings for all lanes, each lane will have its own page.
Ref Clock Termination	
	Lane-specific Reference Clock Impedance setting.

Option	Description
Lane *X* Ref Clock Termination (ohms)	Reference clocks are tied to pairs of lanes in the AC22iHD1000 devices /packages.

Rx PMA Equalization Pages

These pages allow the customization of the PMA equalization settings of the SerDes. Depending upon whether **Enable lane-specific Rx PMA Settings** is selected on the PMA Settings Page (see page 124), there will be either a single page configuring all lanes, or an individual page for each lane. Each page will have identical settings available to the user.

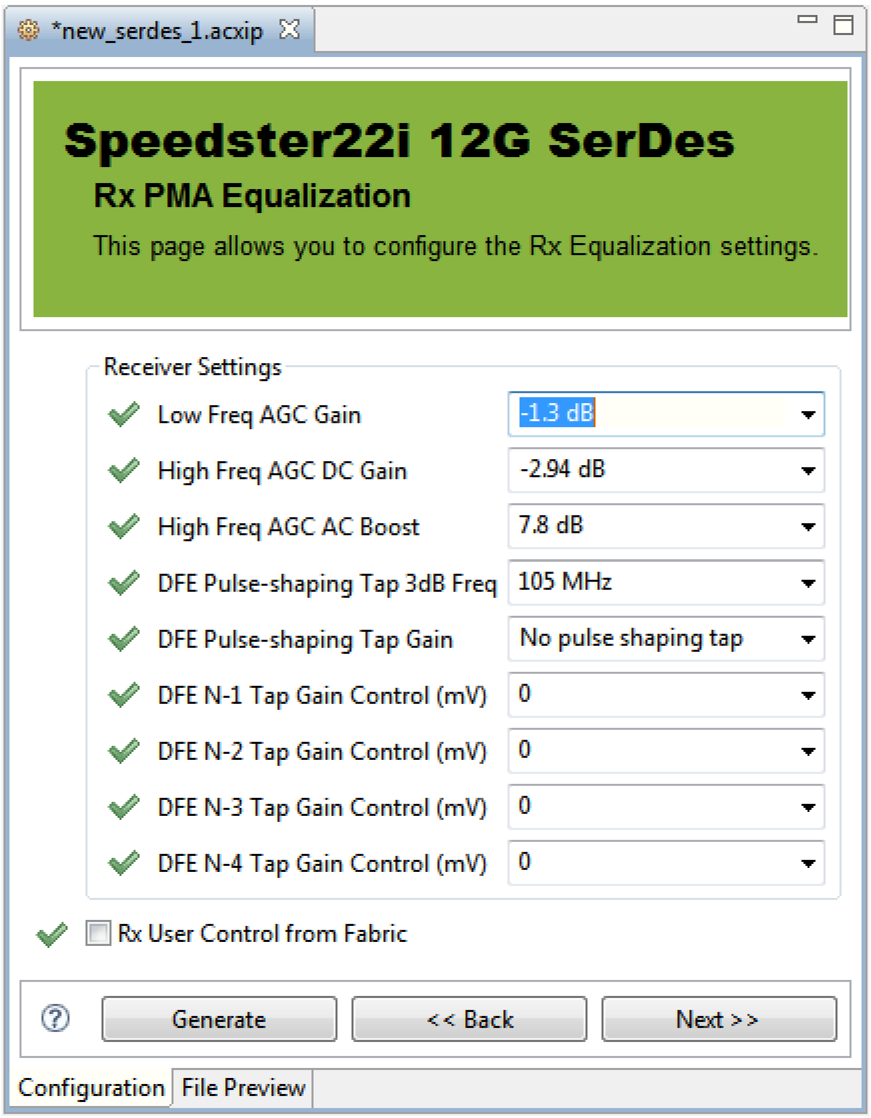


Table 41: *Rx Equalization Page Options*

Options	Description
Receiver Settings	

	Low Freq AGC Gain	Low frequency AGC Gain
	High Freq AGC DC Gain	Rx AGC high frequency DC gain
	High Freq AGC AC Boost	Rx AGC high frequency AC boost
	DFE Pulse-shaping Tap 3dB Freq	DFE pulse-shaping tap 3dB frequency
	DFE Pulse-shaping Tab Gain	DFE pulse-shaping tap gain
	DFE N1 Tap Gain Control (mV)	DFE tap 1 gain control
	DFE N2 Tap Gain Control (mV)	DFE tap 2 gain control
	DFE N3 Tap Gain Control (mV)	DFE tap 3 gain control
	DFE N4 Tap Gain Control (mV)	DFE tap 4 gain control
Rx User Control from Fabric		Control Rx PMA settings from user logic with `chX_i_pma_rxeqlut[32:0]` and `chX_i_pma_rxeqlut_str`.

SerDes Rx PMA PLL Page

This page allows customization of the Rx PLL settings of the SerDes. Depending upon the setting of "Enable lane-specific Rx PMA PLL Settings" on the PMA Settings Page (see page 124), either a single page is provided to configure all lanes, or an individual page is provided for each lane. Regardless of the page title, the available settings are identical.

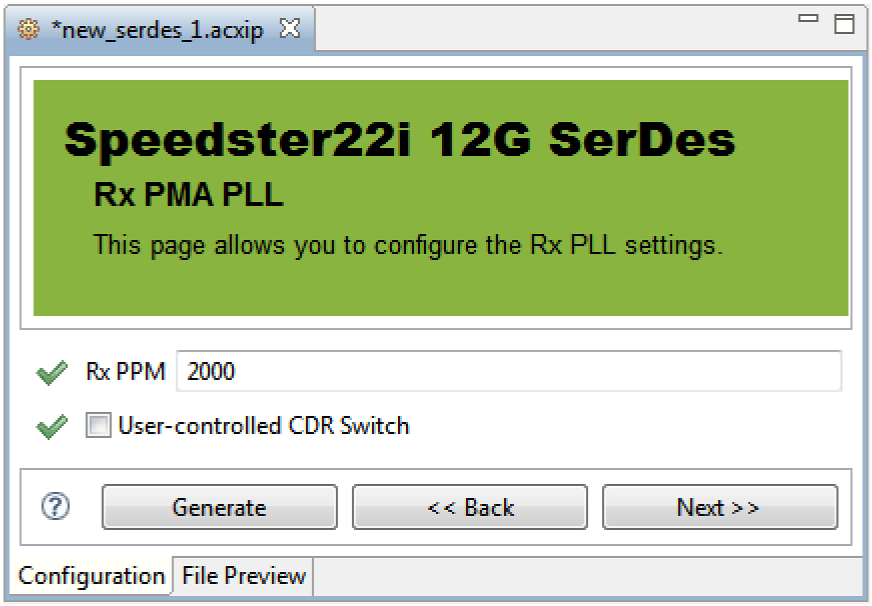


Table 42: *Rx PMA PLL Page Options*

Option	Description
Rx PPM	When Rx PLL is within this PPM range, PMA will consider Tx PLL to be locked
User-controlled CDR Switch	Control CDR PLL to lock to reference or lock to data from fabric

SerDes Tx PMA Driver Page

This page allows the customization of the Tx PMA Driver settings for the SerDes. Depending upon whether the option "Enable lane-specific Tx PMA Driver Settings" is selected on the PMA Settings Page (see page 124), one or more pages with identical available settings will be provided.

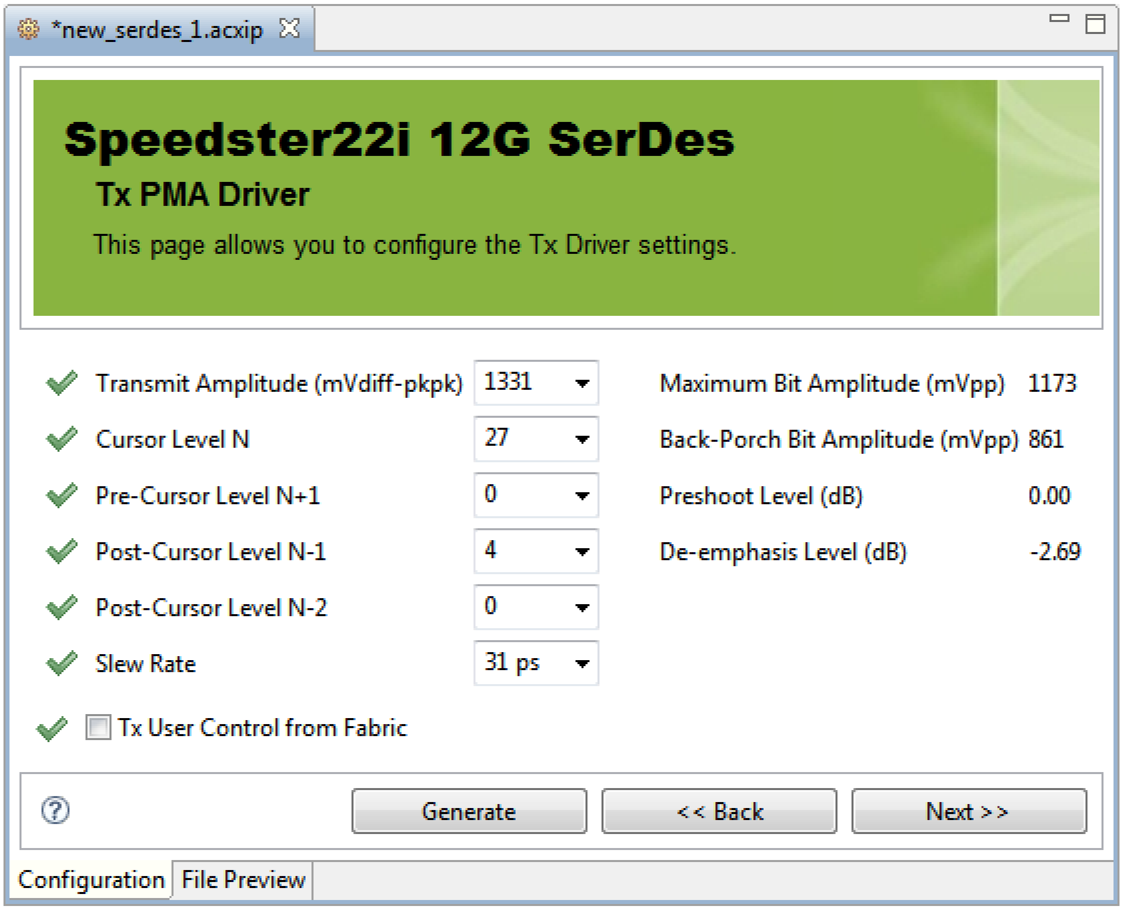


Table 43: *Tx PMA Driver Page Options*

Option	Editable	Description
Transmit Amplitude (mVdiff-pkpk)	Y	Defines the full-scale maximum swing of the driver
Cursor Level N	Y	Defines the total number of driver units allocated to the sum of the driver taps.
Pre-Cursor Level N+1	Y	Defines the total number of driver units allocated to the first pre-cursor
Post-Cursor Level N-1	Y	Defines the total number of driver units allocated to the sum of the first post-cursor tap.
Post-Cursor Level N-2	Y	Defines the total number of driver units allocated to the sum of the second post-cursor tap.
Slew Rate	Y	Tx driver Slew Rate control.
Tx User Control from Fabric	Y	Control PMA Transmit de-emphasis from fabric

Option	Editable	Description
Maximum Bit Amplitude (mVpp)		(DC Amp N) + (DC Amp N+1) + (DC Amp N-1) + (DC Amp N-2)
Back-Porch Bit Amplitude (mVpp)		(DC Amp N) - (DC Amp N+1) - (DC Amp N-1) - (DC Amp N-2)
Preshoot Level (dB)		20 × $\log_{10}$(((DC Amp N) - (DC Amp N+1) - (DC Amp N-1)) ÷ ((DC Amp N) + (DC Amp N+1) - (DC Amp N-1)))
De-emphasis Level (dB)		20 × $\log_{10}$(((DC Amp N) - (DC Amp N+1) - (DC Amp N-1)) ÷ ((DC Amp N) - (DC Amp N+1) + (DC Amp N-1)))

SerDes Tx PMA PLL Page

This page allows the customization of the Tx PMA PLL settings for the SerDes. Depending upon whether the option "Enable lane-specific Tx PMA PLL Settings" is selected on the PMA Settings Page (see page 124), one or more pages with identical available settings will be provided.

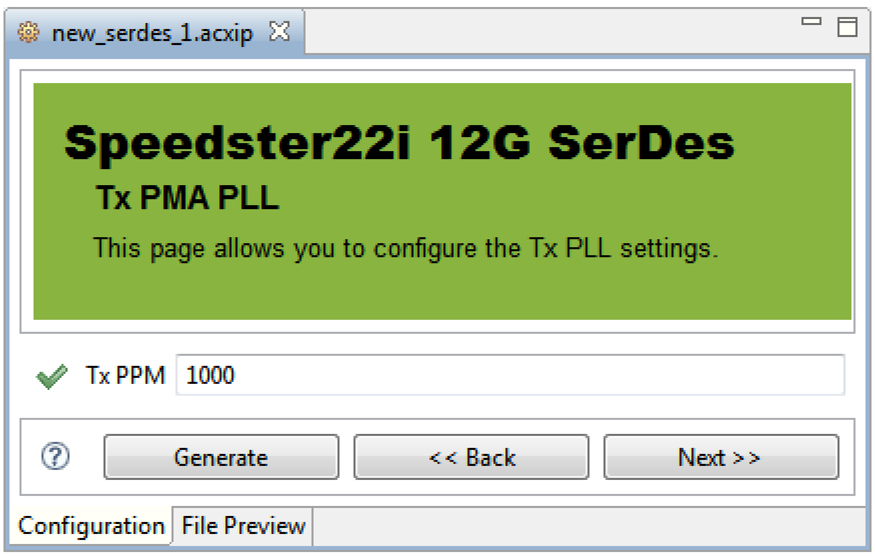


Table 44: *Tx PLL Settings Page Options*

Option	Description
Tx PPM	When Tx PLL is within this PPM range PMA will consider Tx PLL to be locked

PCS Settings Page

The PCS Settings page allows the user to configure the granularity level of Tx/Rx physical coding sublayer (PCS) customization.

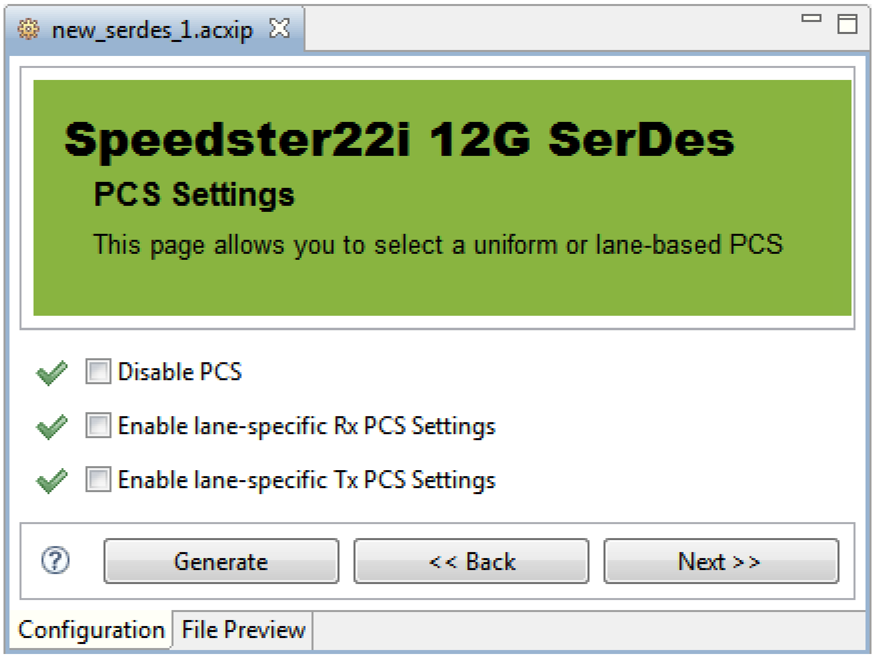


Table 45: *PCS Settings Page Options*

Option	Description
Disable PCS	Disable entire PCS in low-latency bypass mode.
Enable lane-specific Rx PCS Settings	If enabled, instead of a single page configuring the Rx PCS settings and a single page configuring Rx PCS Symbol Alignment for all lanes, each lane will have its own pages.
Enable lane-specific Tx PCS Settings	If enabled, instead of a single page configuring the Tx PCS settings for all lanes, each lane will have its own page.

Rx PCS Settings Page

These pages allow the customization of the Rx PCS settings of the SerDes. Depending upon whether "Enable lane-specific Rx PCS Settings" is selected on the PCS Settings Page (see page 130), there will be either a single page configuring all lanes, or an individual page for each lane. Each page will have identical settings available to the user.

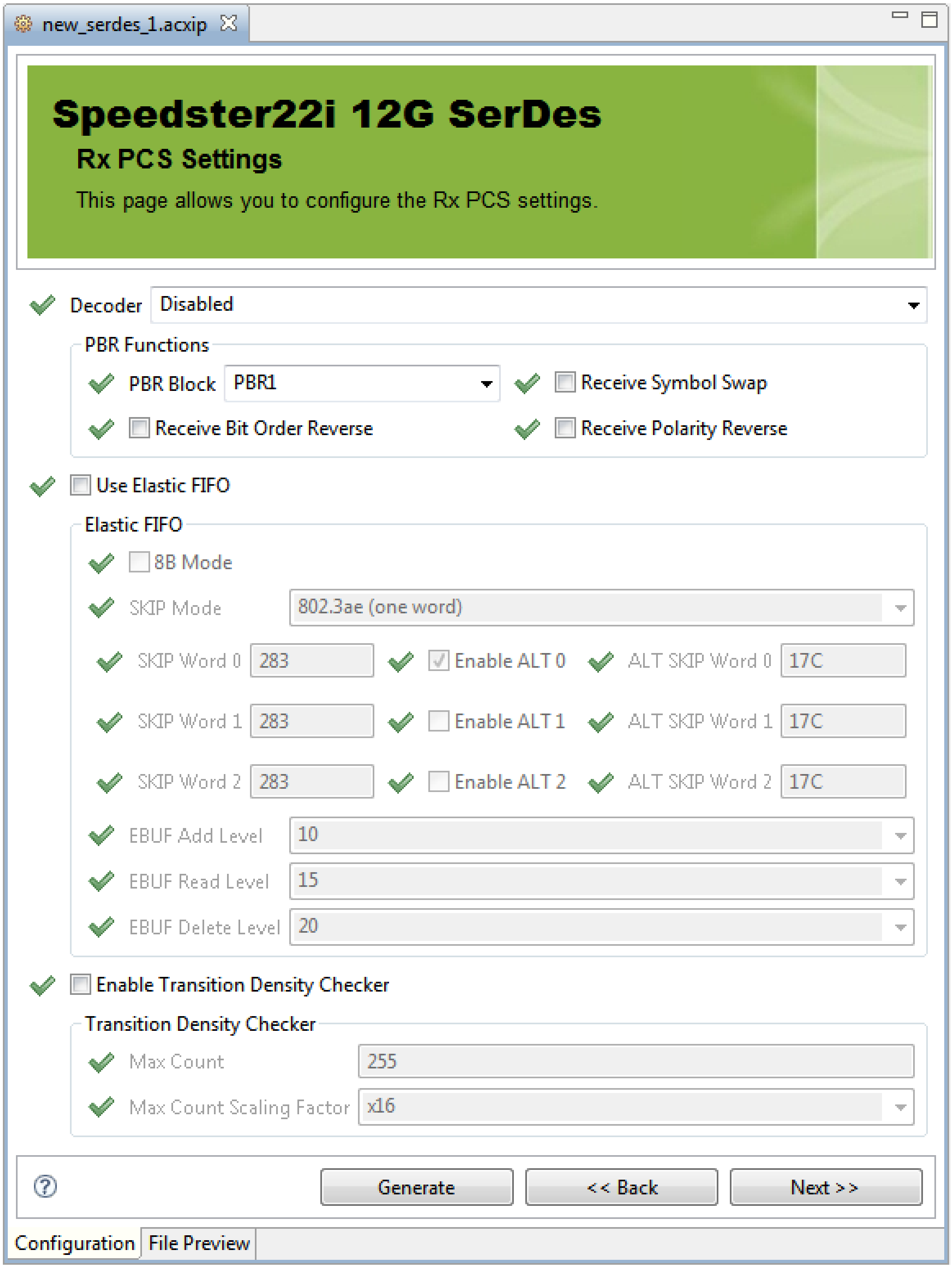

new_serdes_1.acxip
Speedster22i 12G SerDes
Rx PCS Settings
This page allows you to configure the Rx PCS settings.
Decoder
Disabled
PBR Functions
PBR Block
PBR1
Receive Symbol Swap
Receive Bit Order Reverse
Receive Polarity Reverse
Use Elastic FIFO
Elastic FIFO
8B Mode
SKIP Mode
802.3ae (one word)
SKIP Word 0
283
Enable ALT 0
ALT SKIP Word 0
17C
SKIP Word 1
283
Enable ALT 1
ALT SKIP Word 1
17C
SKIP Word 2
283
Enable ALT 2
ALT SKIP Word 2
17C
EBUF Add Level
10
EBUF Read Level
15
EBUF Delete Level
20
Enable Transition Density Checker
Transition Density Checker
Max Count
255
Max Count Scaling Factor
x16
Generate
<< Back
Next >>
Configuration
File Preview

Table 46: *Rx PCS Settings Page Options*

Option		Description
Decoder		PCS Decoder Selection
PBR Functions		
	PBR Block	Rx PBR function block selection
	Receive Symbol Swap	Swap upper word and lower word if in 16 or 20-bit mode
	Receive Bit Order Reverse	Reverse bit order within each word
	Receive Polarity Reverse	Reverse polarity for all bits
Use Elastic FIFO		Enable elastic FIFO
Elastic FIFO		
	8B Mode	Elastic FIFO to operate in 8-bit mode
	SKIP Mode	Specify elastic fifo mode of operation.
	SKIP Word 0	The value to be used as the first SKIP word
	Enable ALT 0	Enable ALT SKIP word 0
	ALT SKIP Word 0	The value to be used as the first ALT SKIP word
	SKIP Word 1	The value to be used as the second SKIP word
	Enable ALT 1	Enable ALT SKIP word 1
	ALT SKIP Word 1	The value to be used as the second ALT SKIP word
	SKIP Word 2	The value to be used as the third SKIP word (used in PCIe mode)
	Enable ALT 2	Enable ALT SKIP word 2 (used in PCIe mode)
	ALT SKIP Word 2	The value to be used as the third ALT SKIP word (used in PCIe mode)
	EBUF Add Level	Elastic FIFO Skip addition level
	EBUF Read Level	Elastic FIFO Read start level
	EBUF Delete Level	Elastic FIFO Skip deletion level

Option		Description
Enable Transition Density Checker		Enable Transition Density Checker
Transition Density Checker		
	Max Count	Count to be multiplied by the scaling factor to trigger transition density error flag
	Max Count Scaling Factor	Scaling factor to multiply the max count to trigger transition density error flag

Rx PCS Symbol Alignment Page

These pages allow the customization of the Rx PCS Symbol Alignment settings of the SerDes. Depending upon whether "Enable lane-specific Rx PCS Settings" is selected on the PCS Settings Page (see page 130), there will be either a single page configuring all lanes, or an individual page for each lane. Each page will have identical settings available to the user.

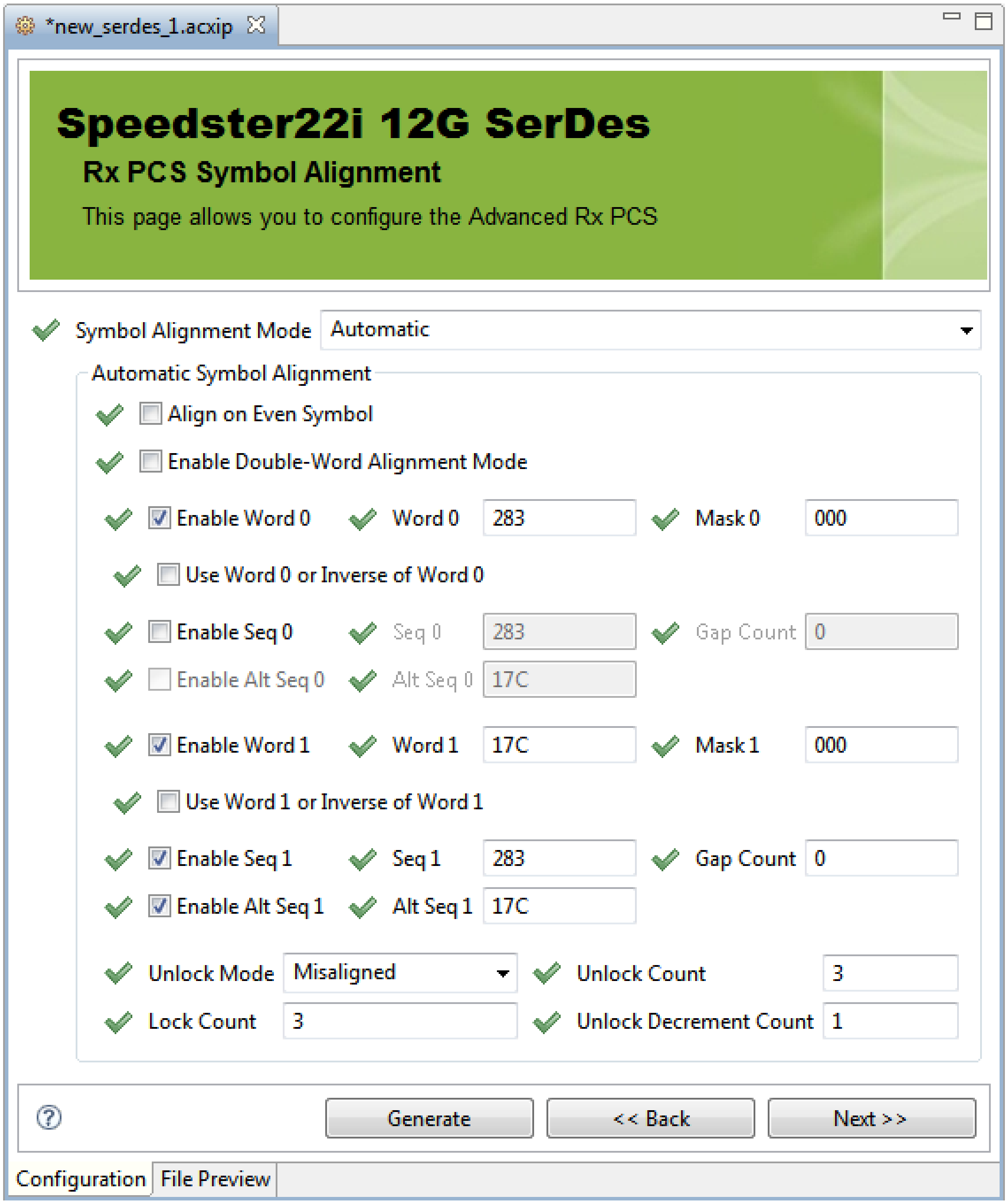


Table 47: *Rx PCS Symbol Alignment Page Options*

Option		Description
Symbol Alignment Mode		Symbol alignment modes
Automatic Symbol Alignment		
	Align on Even Symbol	Align word 0 on even symbol only

	Enable Double-Word Alignment Mode	Double-word alignment mode (word0 && word1)
	Enable Word 0	Enable word 0 to use for symbol alignment
	Word 0	Word 0 for symbol alignment
	Mask 0	Mask for word 0 for symbol alignment
	Use Word 0 or Inverse of Word 0	Use either word 0 or inverse of word 0 as alignment character
	Enable Seq0	Enable Seq 0 word for symbol alignment
	Seq 0	Seq 0 Word
	Gap Count	Word and Seq0/Alt Seq0 Gap Count
	Enable Alt Seq 0	Enable Alternate Seq 0 word for symbol alignment
	Alt Seq 0	Alternate Seq 0 Word
	Enable Word 1	Enable word 1 to use for symbol alignment
	Word 1	Word 1 for symbol alignment
	Mask 1	Mask for word 1 for symbol alignment
	Use Word 1 or Inverse of Word 1	Use either word 1 or inverse of word 1 as alignment character
	Enable Seq 1	Enable Seq 1 word for symbol alignment
	Seq 1	Seq 1 Word
	Gap Count	Word and Seq1/Alt Seq1 Gap Count
	Enable Alt Seq 1	Enable Alternate Seq 1 word for symbol alignment
	Alt Seq 1	Alternate Seq 1 Word
	Unlock Mode	Condition to consider for symbol alignment unlock
	Unlock Count	Number of unlock count to unlock the alignment
	Lock Count	Number of successful matches to lock the alignment
	Unlock Decrement Count	Number of errors to consider to increment unlock count

Tx PCS Settings Page

These pages allow the customization of the Tx PCS settings of the SerDes. Depending upon whether "Enable lane-specific Tx PCS Settings" is selected on the PCS Settings Page (see page 130), there will be either a single page configuring all lanes, or an individual page for each lane. Each page will have identical settings available to the user.

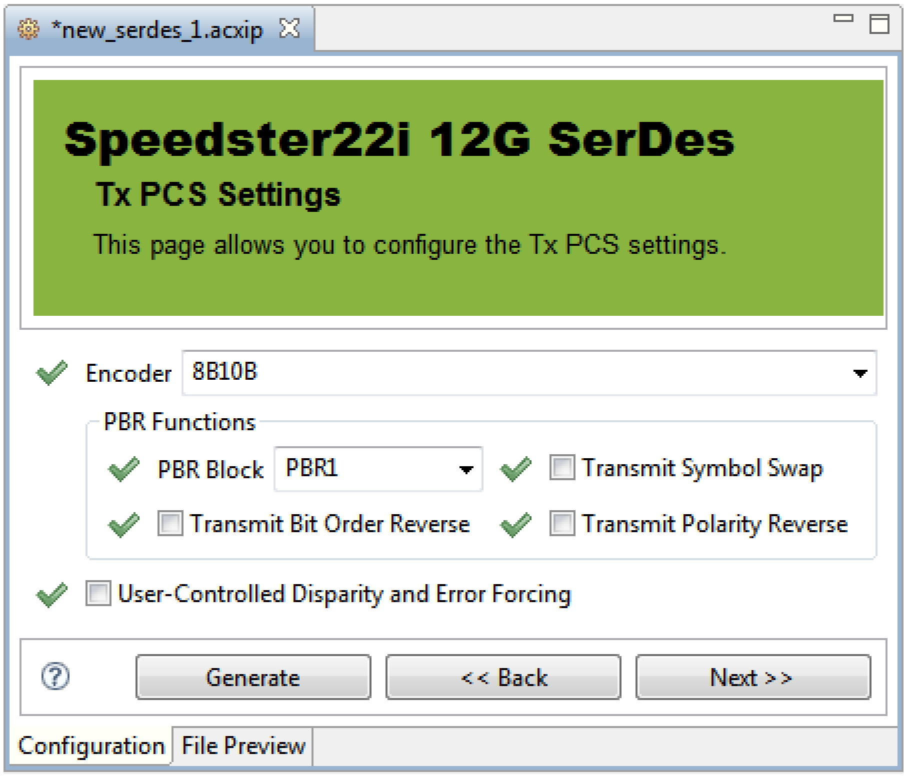


Table 48: *Tx PCS Settings Page Options*

Option		Description
Encoder		PCS Encoder Selection
PBR Functions		
	PBR Block	Tx PBR function block selection
	Transmit Symbol Swap	Swap upper word and lower word if in 16 or 20-bit mode
	Transmit Bit Order Reverse	Reverse bit order within each word
	Transmit Polarity Reverse	Reverse polarity for all bits
User-Controlled Disparity and Error Enforcing		Control disparity and error forcing from fabric

Channel Bonding Page

This page allows the customization of the Channel Bonding settings of the SerDes.

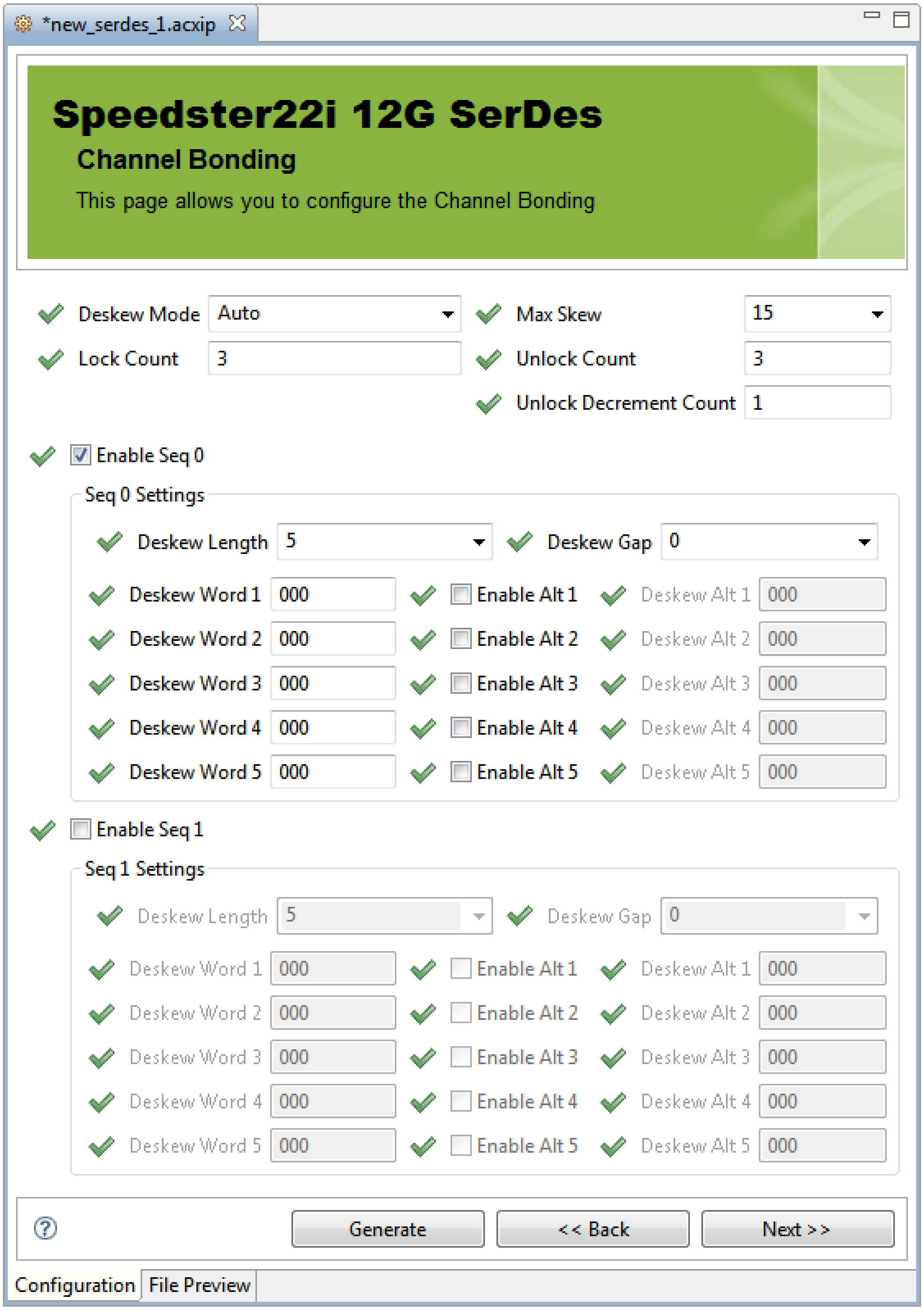

*new_serdes_1.acxip
Speedster22i 12G SerDes
Channel Bonding
This page allows you to configure the Channel Bonding
Deskew Mode
Auto
Max Skew
15
Lock Count
3
Unlock Count
3
Unlock Decrement Count
1
Enable Seq 0
Seq 0 Settings
Deskew Length
5
Deskew Gap
0
Deskew Word 1
000
Enable Alt 1
Deskew Alt 1
000
Deskew Word 2
000
Enable Alt 2
Deskew Alt 2
000
Deskew Word 3
000
Enable Alt 3
Deskew Alt 3
000
Deskew Word 4
000
Enable Alt 4
Deskew Alt 4
000
Deskew Word 5
000
Enable Alt 5
Deskew Alt 5
000
Enable Seq 1
Seq 1 Settings
Deskew Length
5
Deskew Gap
0
Deskew Word 1
000
Enable Alt 1
Deskew Alt 1
000
Deskew Word 2
000
Enable Alt 2
Deskew Alt 2
000
Deskew Word 3
000
Enable Alt 3
Deskew Alt 3
000
Deskew Word 4
000
Enable Alt 4
Deskew Alt 4
000
Deskew Word 5
000
Enable Alt 5
Deskew Alt 5
000
Generate
<< Back
Next >>
Configuration
File Preview

Table 49: *Channel Bonding Settings Page Options*

Option		Description
Deskew Mode		Deskew operation mode: automatic, manual, or bit-slip.
Max Skew		Maximum skew between lanes
Lock Count		Number of successful matches to lock the deskew logic
Unlock Count		Number of unlock count to unlock deskew logic
Unlock Decrement Count		Number of errors to increment unlock count
Enable Seq 0		Enable deskew sequence 0
Seq 0 Settings		
	Deskew Length	Number of deskew words for lane deskew operation on Seq 0
	Deskew Gap	Number of don't care symbols between deskew sequences on Seq 0
	Deskew Word 1	Deskew Sequence 0 word 1
	Enable Alt 1	Enable deskew sequence 0 alternate word 1
	Deskew Alt 1	Deskew Sequence 0 Alternate word 1
	Deskew Word 2	Deskew Sequence 0 word 2
	Enable Alt 2	Enable deskew sequence 0 alternate word 2
	Deskew Alt 2	Deskew Sequence 0 Alternate word 2
	Deskew Word 3	Deskew Sequence 0 word 3
	Enable Alt 3	Enable deskew sequence 0 alternate word 3
	Deskew Alt 3	Deskew Sequence 0 Alternate word 3
	Deskew Word 4	Deskew Sequence 0 word 4
	Enable Alt 4	Enable deskew sequence 0 alternate word 4
	Deskew Alt 4	Deskew Sequence 0 Alternate word 4
	Deskew Word 5	Deskew Sequence 0 word 5

Option		Description
	Enable Alt 5	Enable deskew sequence 0 alternate word 5
	Deskew Alt 5	Deskew Sequence 0 Alternate word 5
Enable Seq 1		Enable deskew sequence 1
Seq 1 Settings		
	Deskew Word 1	Deskew Sequence 1 word 1
	Enable Alt 1	Enable deskew sequence 1 alternate word 1
	Deskew Alt 1	Deskew Sequence 1 Alternate word 1
	Deskew Word 2	Deskew Sequence 1 word 2
	Enable Alt 2	Enable deskew sequence 1 alternate word 2
	Deskew Alt 2	Deskew Sequence 1 Alternate word 2
	Deskew Word 3	Deskew Sequence 1 word 3
	Enable Alt 3	Enable deskew sequence 1 alternate word 3
	Deskew Alt 3	Deskew Sequence 1 Alternate word 3
	Deskew Word 4	Deskew Sequence 1 word 4
	Enable Alt 4	Enable deskew sequence 1 alternate word 4
	Deskew Alt 4	Deskew Sequence 1 Alternate word 4
	Deskew Word 5	Deskew Sequence 1 word 5
	Enable Alt 5	Enable deskew sequence 1 alternate word 5
	Deskew Alt 5	Deskew Sequence 1 Alternate word 5

BIST Page

The BIST page of the Speedster22i SerDes Configuration Editor (see page 120) allows the user to choose whether the SerDes lanes will each have individual BIST configurations, or will share a single configuration.

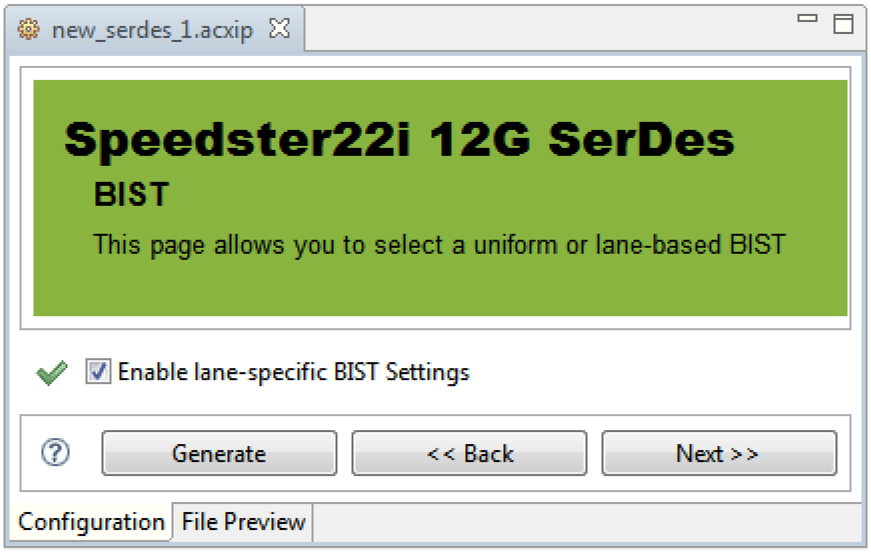


Table 50: ***SerDes Editor BIST Options***

Option	Description
Enable lane-specific BIST Settings	When enabled, each lane will have its own page of BIST settings. When disabled, all lanes will share a single page of BIST settings.

BIST Settings Page

These pages allow the customization of the BIST settings of the SerDes. Depending upon whether "Enable lane-specific BIST Settings" is selected on the BIST Page (see page 140), there will be either a single page configuring all lanes, or an individual page for each lane. Each page will have identical settings available to the user.

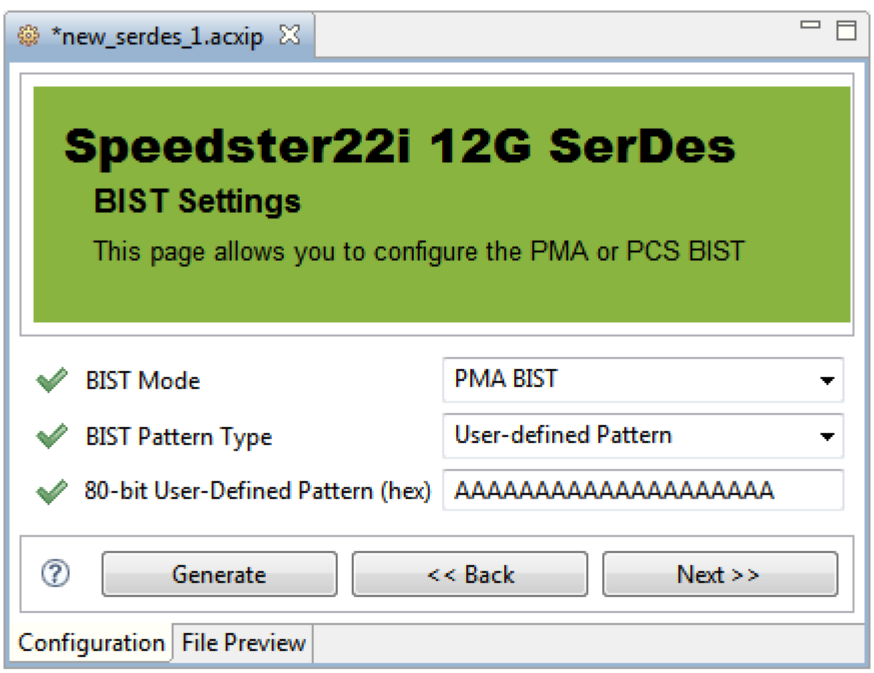


Table 51: ***BIST Settings Page Options***

Option	Description
BIST Mode	Choose whether to use PMA BIST, PCS BIST, or neither (disabled)
BIST Pattern Type	Choose which BIST pattern should be used. The available PRBS patterns will vary by mode.
80-bit User-Defined Pattern (hex)	Enabled only when "BIST Pattern Type" is set to **User-defined Pattern**, the user may enter any 80-bit pattern in hex.

Advanced Page

This page has no options of its own. It is only used to provide separation between the usual editor settings and the pages of advanced settings, including the Link Tuning Page (see page 146) and Register Settings page(s) (see page 157).

Link Tuning Parameters Page

This page is used to configure the link tuning parameters used for PMA receive auto equalization and eye measurements. These settings are then reflected in the Link Tuning Page (see page 146).

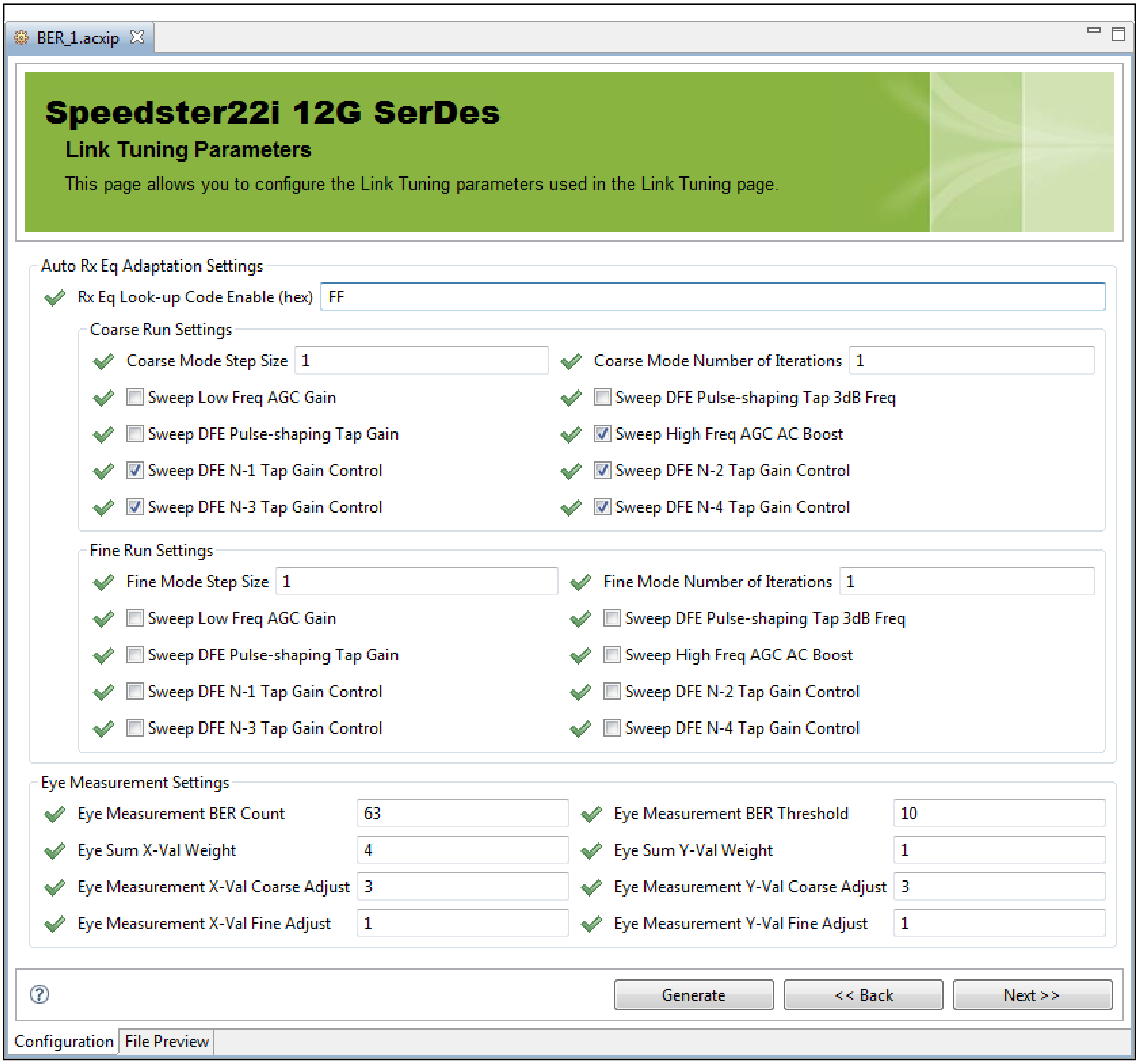


Figure 11: ***Link Tuning Parameters Page***

The table below lists the various option setting for the page.

Table 52: ***Options for Link Tuning Parameters Page***

Option	Global	Editable	Description
Auto Rx Eq Adaption Settings			
Rx Eq Look-up Code Enable (hex)		✓	Sets the equalization calibration lookup table code. When in lookup table mode, controls whether or not the associated lookup code is included. Lookup codes 1-7 are predefined, lookup code 0 is defined by *_LUP0_NT_ inputs.
Coarse Run Settings			
Coarse Mode Step Size		✓	Sets the receive equalization calibration coarse mode step size. When swept, each variable starts from 0 and is increased by this value until saturated.
Coarse Mode Number of Iterations		✓	Sets the equalization calibration coarse iteration count, or the number of iterations performed during coarse mode.
Sweep Low Freq AGC Gain		✓	Enables sweeping of the low-frequency AGC gain parameter during receive auto-equalization tuning.
Sweep DFW Pulse-shaping Tap 3dB Freq		✓	Enables sweeping of the DFE pulse-shaping tap 3-dB frequency parameter during receive auto-equalization tuning,
Sweep DFE Pulse-shaping Tap Gain		✓	Enables sweeping of the low-frequency AGC gain parameter during receive auto-equalization tuning.
Sweep High Freq AGC AC Boost		✓	Enables sweeping of the high-frequency AGC AC boost parameter during receive auto-equalization tuning.
Sweep DFE N-[*1..Number of Lanes*] Tap Gain Control		✓	Enable sweeping of the DFE N-<*lane number*> tap gain control parameter during receive auto-equalization tuning
Fine Run Settings			
Fine Mode Step Size		✓	RX Equalization calibration fine mode step size. When swept, each variable starts from 0, and is increased by this value until saturated.
Fine Mode Number of Iterations		✓	Equalization calibration coarse iteration count control. Sets the number of iterations performed during fine mode.

Option	Global	Editable	Description
Sweep Low Freq AGC Gain		✓	Enable sweeping of low-frequency AGC gain parameter during receive auto-equalization tuning
Sweep DFW Pulse-shaping Tap 3dB Freq		✓	Enable sweeping of DFE pulse-shaping tap 3dB frequency parameter during receive auto-equalization tuning
Sweep DFE Pulse-shaping Tap Gain		✓	Enables sweeping of the low-frequency AGC gain parameter during receive auto-equalization tuning.
Sweep High Freq AGC AC Boost		✓	Enables sweeping of the high-frequency AGC AC Boost parameter during receive auto-equalization tuning,
Sweep DFE N-[1..4] Tap Gain Control		✓	Enable sweeping of the DFE N−[1..4] tap gain control parameter during receive auto-equalization tuning,
Eye Measurement Settings			
Eye Measurement BER Count		✓	During 4-point eye measurement, sets the maximum number of samples to be taken when determining the eye boundary.
Eye Measurement BER Threshold		✓	After the sample count set in BER Count is reached, the BER threshold value defines the boundary of the eye. When the number of bit-errors out of the samples exceed the BER threshold, the boundary has been reached.
Eye Sum X-Val Weight		✓	Sets the X/Y weighting values used when calculating weighted sum of all sweeps. The Y weight may need to be increased to account for stressful channel loss. The X weight may need to be increased if jitter is an issue.
Eye Sum Y-Val Weight		✓	
Eye Measurement X-Val Coarse Adjust		✓	Sets the 4-point eye sweep coarse increment values. The FSM increases the absolute X/Y values by *_COARSE_NT_ Setting.†
Eye Measurement Y-Val Coarse Adjust		✓	
Eye Measurement X-Val Fine Adjust		✓	Sets the 4-point eye sweep fine increment values. The FSM increases the absolute X/Y values by *_FINE_NT_ Setting.†
Eye Measurement Y-Val Fine Adjust		✓	

† Setting **Y-Val Coarse** and **Y-Val Fine** to **0** will cause the FSM to sweep along the X axis only, using **X-Val Coarse** and **X-Val Fine**.

Link Tuning Page

> A complete GUI Link Tuning reference is available as a separate document, the *SerDes Link Tuning GUI User Guide*. Other reference documents will be available for alternate (non-GUI) automatic link tuning procedures.
>
> Within this ACE User Guide, additional important details about the Link Tuning process may be found in the section Live Link Tuning for SerDes and Derived Interfaces (see page 390).

Unlike the other pages of the IP Configuration Editors, the Link Tuning Page is meant to interact with a live FPGA using JTAG (through the Bitporter). It allows the user to observe and tune IP configuration settings of a live SerDes link. Interacting with a live link allows the configuration to be tweaked according to the specific details of the physical interface with devices external to the FPGA.

This page allows observation of SerDes PMA status, Figure of Merit, and Rx PMA Eq / Tx PMA Driver parameters on a live FPGA / board. Only the lanes active in the `.acxip` configuration will be observed. This page also allows Rx Auto Eq Tuning across all the active lanes with a simple button press.

Once auto-tuning is complete, the "PMA Status" table will be updated to show the latest per-lane status, and the "PMA Rx Equalization" table will show the updated Figure Of Merit values and updated Rx Eq and Tx Driver settings for each lane. Any Rx Eq or Tx Driver values queried from the chip found to be different from the values in the IP Configuration Editor will be highlighted in yellow in the appropriate "Chip" column field.

When values queried from the chip are found to be different (highlighted yellow) from the ones stored in the IP Configuration Editor, the **Sync GUI with Chip** button may be pressed to automatically capture all the new (yellow) auto-tuned values from the chip to the IP Configuration Editor (where they can be saved to the current `.acxip` file).

The number of columns of information in the tables of this page will vary according to number of SerDes lanes active in the current IP configuration. (Each lane will get its own uniquely configured column of information in the Status and Equalization tables.)

The top of the page shows "PMA Status" information queried from the chip, with LED icons to show the individual status values for each lane. Green icons represent a good/high status, red icons represent a bad/low status, and grey icons indicate a lack of data (when the status has not yet been retrieved).

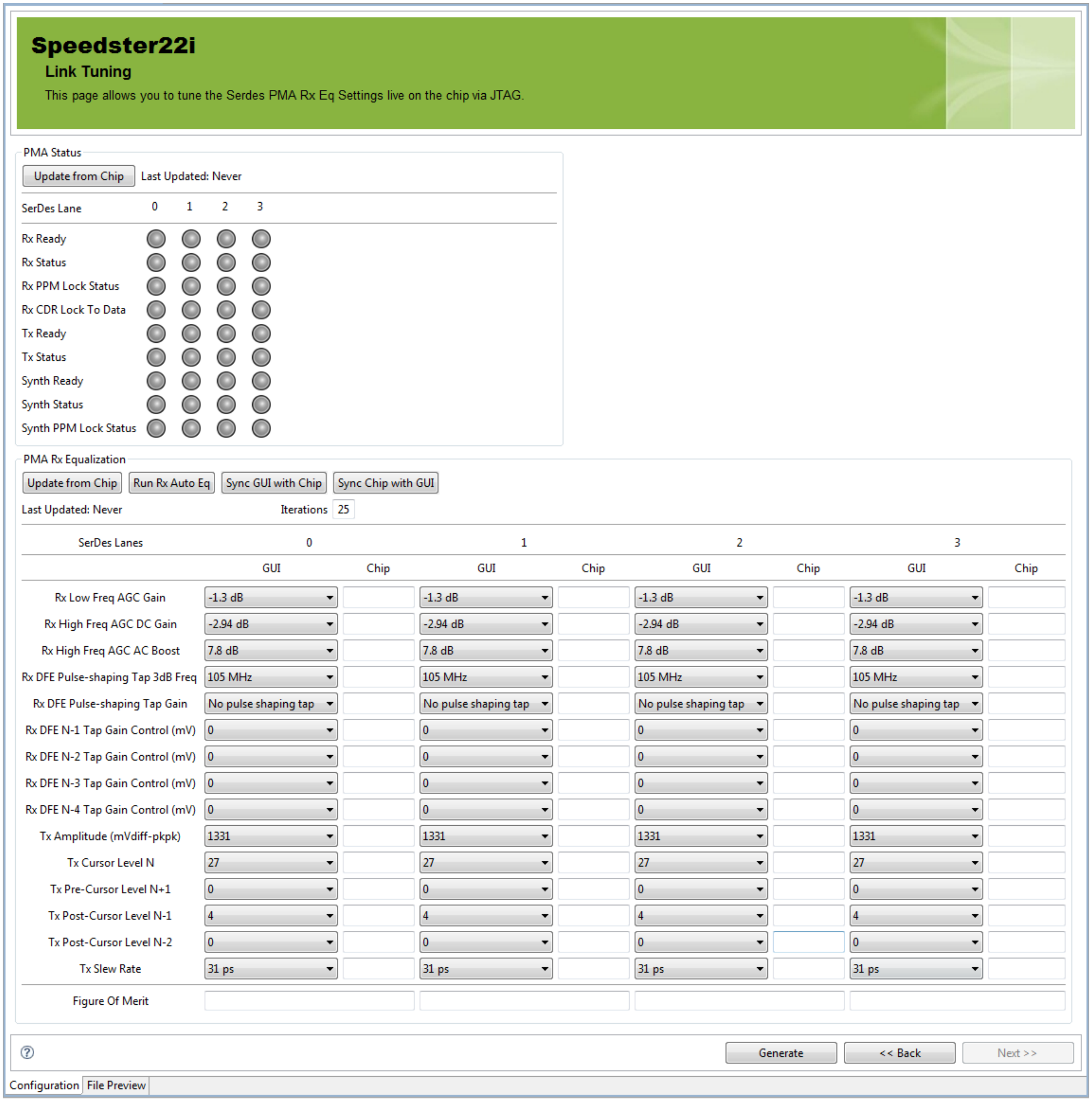


Figure 12: Example screenshot of a four-lane Link Tuning page

The lower portion of the page includes the "PMA Rx Equalization" table, with the calculated **Figure Of Merit** and the various configuration settings for each active lane. Each lane includes an editable column for the current "GUI" settings (representing the settings in the IP Configuration Editor, i.e. the `.acxip` file), and a read-only column of "Chip" settings which were retrieved from the live chip. The values in the Chip column will be highlighted in yellow when they differ from the GUI settings.

Action	Description
Update from Chip	Queries the chip for the PMA Status, Eq config, and **Figure Of Merit** values for all active Lanes. The **Last Updated** timestamp for both tables will be set according to when the values were returned from the chip. Any Eq values queried from the chip found to be different from the values in the associated "GUI" column field will be highlighted in yellow in the appropriate "Chip" column field. **Update from Chip** may safely be run at any time, including when live data is flowing through the interface. It will not introduce bit errors.
Run Rx Auto Eq	Runs **Iterations** passes of an Rx Eq auto-tuning algorithm on the chip for all active lanes. When complete, the "PMA Status" table will be updated to show the latest per-lane status, and the "PMA Rx Equalization" table will show the updated **Figure Of Merit** values and updated Eq settings for each lane. Any Eq values queried from the chip found to be different from the values in the associated "GUI" column field will be highlighted in yellow in the appropriate "Chip" column field. Note that running Rx Auto Eq will not succeed if the initial FOM is 0, and sometimes might fail if the initial FOM is very low. If Rx Auto Eq cannot be run successfully, users may need to manually tweak the Rx Eq settings until a non-zero initial FOM is achieved. **Run Rx Auto Eq** is not supported during live traffic. Changing any Rx Eq parameter (including those changed during auto-tuning) while receiving live traffic will result in bit errors.
Sync GUI with Chip	Copies config values from the "Chip" column into the "GUI" column for all active lanes. These altered config values will also appear in the other appropriate pages of the IP Configuration Editor, and will be saved as part of the current `.acxip` file.
Sync Chip with GUI	Sends the current configuration (as shown in the "GUI" column) to the Chip, and then performs **Update from Chip.**

Option	Global	Editable	Description
PMA Status			
Last Updated	✔		A timestamp from the last time the GUI was updated with data from the chip.
Rx Ready			Rx lane ready status signal Red: Rx lane is not ready for data transmission Green: Rx lane is ready for data transmission
Rx Status			Rx lane state transition status. Indicates the Rx has completed a requested state transition. Red: Rx lane has not completed its state change Green: Rx lane has completed its state change

Option	Global	Editable	Description
Rx PPM Lock Status			Rx lane PLL lock status Green: CDR VCO is locked to reference clock
Rx CDR Lock To Data			Rx CDR Lock to Data status indicator Red: CDR is locked to reference clock Green: CDR is locked to data
Tx Ready			Tx lane ready status signal Red: Tx lane is not ready for data transmission Green: Tx lane is ready for data transmission
Tx Status			Tx lane state transition status. Indicates the Tx has completed a requested state transition. Red: Tx lane has not completed its state change Green: Tx lane has completed its state change
Synth Ready			Synth lane ready status signal. Red: Synth is not ready for data transmission Green: Synth is ready for data transmission
Synth Status			Synth lane state transition status. Indicates when the Synth has completed a requested state transition. Red: Synth has not completed its state change Green: Synth has completed its state change
Synth PPM Lock Status			Synth lane PLL lock status Green: Synth VCO is locked to reference clock
PMA Rx Equalization			
Last Updated	✓		A timestamp from the last time the GUI was updated with data from the chip.
Iterations	✓	✓	The number of iterations the user wishes the FOM capture (during **Update from Chip**) and Rx Auto Eq tuning (during **Run Rx Auto Eq**) algorithms to be executed.
Rx Low Freq AGC Gain		✓	Rx AGC low frequency gain
Rx High Freq AGC DC Gain		✓	Rx AGC high frequency gain
Rx High Freq AGC AC Boost		✓	Rx AGC high frequency AC boost

Option	Global	Editable	Description
Rx DFE Pulse-shaping Tap 3dB Freq		✔	Rx DFE pulse-shaping tap 3dB frequency
Rx DFE Pulse-shaping Tap Gain		✔	Rx DFE pulse-shaping tap gain
Rx DFE N-[*1..4*] Tap Gain Control (mV)		✔	Rx DFE tap [*1..4*] gain control
Tx Amplitude (mVdiff-pkpk)		✔	Defines the full-scale maximum swing of the Tx driver
Tx Cursor Level N		✔	Defines the total number of driver units allocated to the sum of the driver taps
Tx Cursor Level N+1		✔	Defines the total number of driver units allocated to the first pre-cursor
Tx Cursor Level N-1		✔	Defines the total number of driver units allocated to the sum of the first post-cursor tap
Tx Cursor Level N-2		✔	Defines the total number of driver units allocated to the sum of the second post-cursor tap
Tx Slew Rate		✔	Tx driver Slew Rate control
Figure Of Merit			A Figure Of Merit (or FOM) value represents the quality of the signal observed on the Rx side of the PMA by using a 4-point eye analysis. The FOM value is shown as an integer value instead of as a diagram. The higher the FOM integer value, the better the signal quality (or eye opening) is. The FOM value is captured using the number of iterations specified, and returns the average value across iterations. The FOM can change slightly between each capture as the PMA sees different corners in the data pattern. Note that FOM is only captured if all SerDes PMA status signals are high (green LEDs), otherwise FOM cannot be captured.

Link Characterization Parameters Page

This page is used to set the value for parameters to be used for PMA receiver equalizer and transmitter drive settings to capture BER values to characterize the link. During characterization, each of the possible combinations of settings defined under "Parameters to Sweep" are used.

The settings here are reflected on the Link Characterization Page (see page 153).

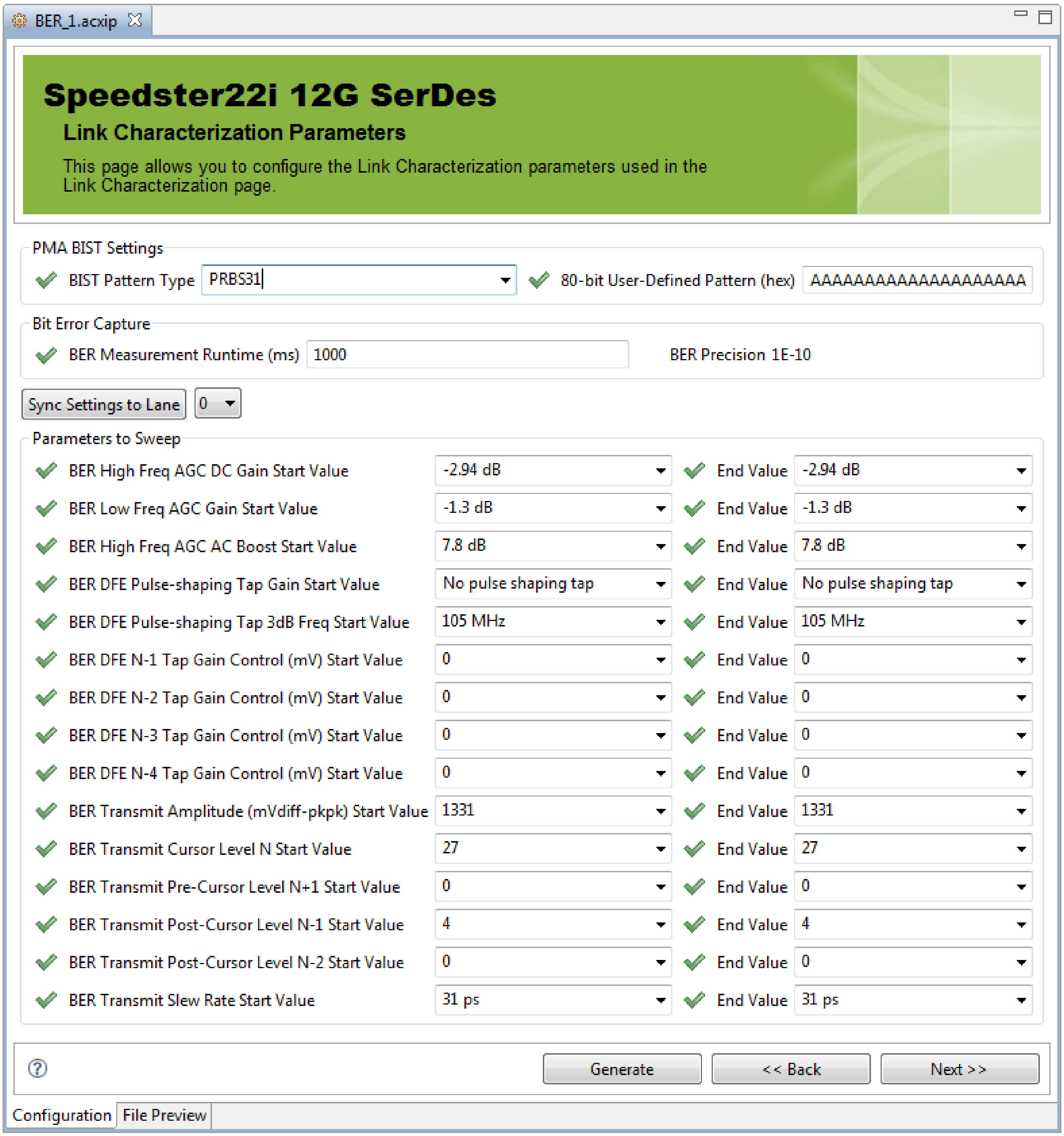


Figure 13: ***Link Characterization Parameters Page***

This page has a single action, listed in the table below.

Table 53: ***Link Characterization Parameters Page Actions***

Action	Description
Sync Settings to Lane	Clicking the button writes the set parameters to the device (the pull-down is used to select the lane).

The various options for this page are listed in the table below.

Table 54: ***Link Characterization Parameters Page Options***

Options	Global	Editable	Description
PMA BIST Settings			
BIST Pattern Type		✔	Pull-down to select BIST pattern type. Options are PRBS7, PRBS23, PRBS31 (default), and User-defined Pattern.
80-bit User-Defined Pattern (hex)		✔	Sets the user-defined BIST pattern if User-defined Pattern is selected in BIST Pattern Type.
Bit Error Capture			
BER Measurement Runtime (ms)		✔	Sets the window of time (in milliseconds) to capture bit errors. The BER precision is the result of this time and the receive data rate.
BER Precision			Displays the resulting BER precision based on BER Measurement Runtime. Precision = Receive Data Rate (Gbps)/runtime (ms)
Parameters to Sweep†			
BER High Freq AGC DC Gain – Start Value/End Value		✔	Sets the amount of receive high-frequency AGC DC gain.
BER Low Freq AGC Gain – Start Value/End Value		✔	Sets the amount of receive low-frequency AGC gain.
BER High Freq AGC AC Boost – Start Value/End Value		✔	Sets the amount of receive high-frequency AGC AC boost.
BER DFE Pulse-shaping Tap Gain – Start Value/End Value		✔	Sets the amount of receive DFE pulse-shaping tap gain.
BER DFE Pulse-shaping Tap 3dB Freq – Start Value /End Value		✔	Sets the amount of receive DFE pulse-shaping tap 3dB frequency.

Options	Global	Editable	Description
BER DFE N-[1..4] Tap Gain Control (mV) – Start Value /End Value		✔	Sets the amount of receive DFE tap gain control.
BER Transmit Amplitude (mFdiff-pkpk) – Start Value /End Value		✔	Sets the full-scale maximum swing of the transmit driver
BER Transmit Cursor Level N – Start Value/End Value		✔	Defines the total number of driver units allocated to the sum of the driver taps.
BER Transmit Cursor Level N+1 – Start Value/End Value		✔	Defines the total number of driver units allocated to the sum of first pre-cursor tap.
BER Transmit Cursor Level N-1 – Start Value/End Value		✔	Defines the total number of driver units allocated to the sum of the first post-cursor tap.
BER Transmit Cursor Level N-2 – Start Value/End Value		✔	Defines the total number of driver units allocated to the sum of the second post-cursor tap.
BER Transmit Slew Rate – Start Value/End Value		✔	Sets the transmit driver slew rate.
† Sweeps from **Start Value** to **End Value**. Setting the **Start Value** equal to the **End Value** fixes the parameter to a single value for all sweeps.			

Link Characterization Page

This page used to run and control exhaustive sweeps over various PMA receiver equalizer and transmitter drive settings to capture BER values to characterize the link.

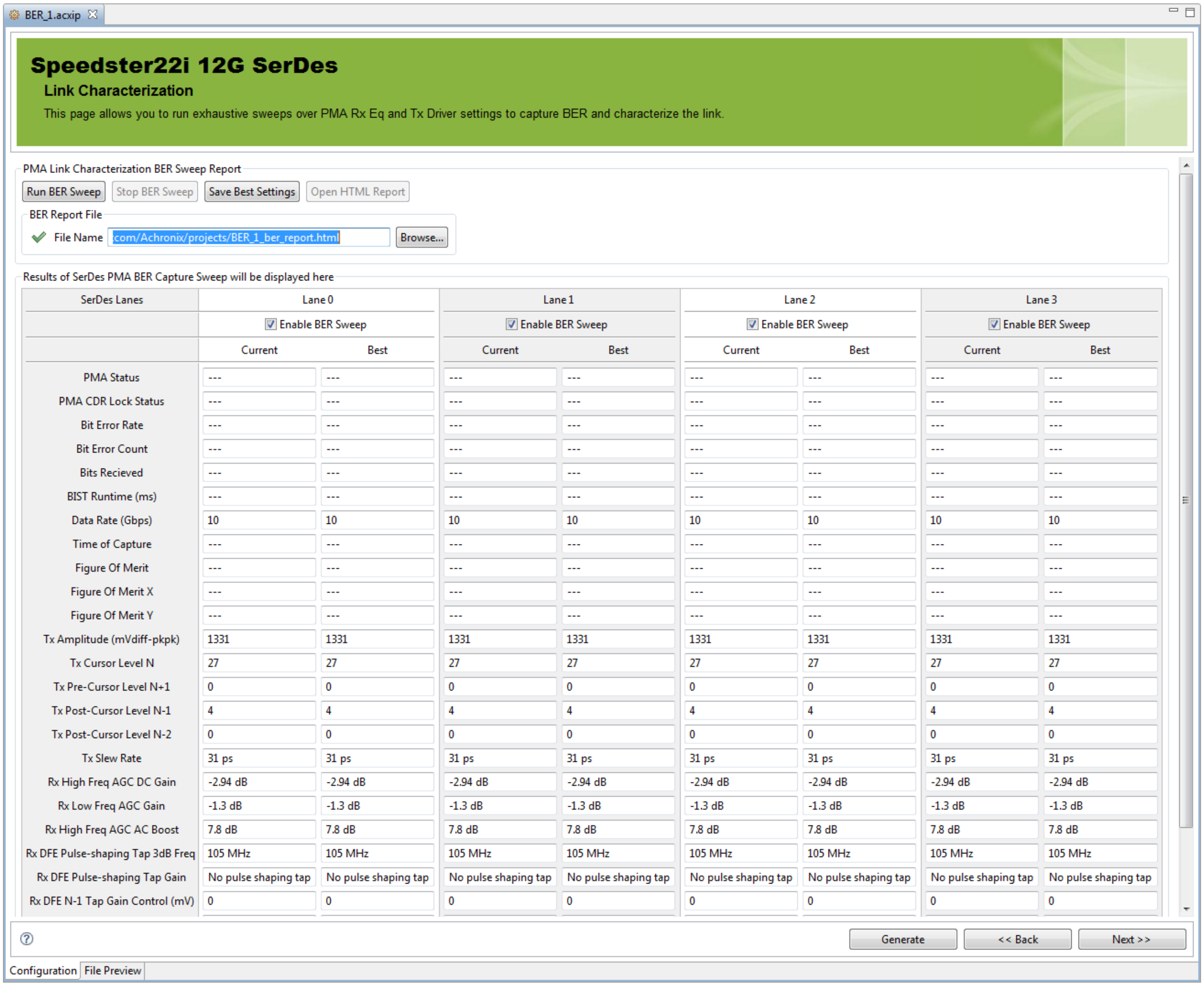


Figure 14: ***Link Characterization Page***

This page has five actions/options, listed in the table below.

Table 55: *PMA Link Characterization BER Sweep Report Actions*

Action	Description
Run BER Sweep	Starts the BER sweep over the values set via the Link Characterization Parameters Page (see page 150) For each set of parameters, runs the PMA BIST. For each sweep, the BER, Figure of Merit, PMA lock status, etc., is captured and stored. The results of each sweep are displayed in the GUI as a sweep ends. If open, the HTML report also refreshes with each sweep.
Stop BER Sweep	Stops the characterization at the end of the current sweep.

Action	Description
Save Best Settings	For each lane, finds the best receive and transmit settings and saves these settings to the SerDes configuration for generating an optimized bitstream. The first group of settings that achieved zero BER, or with the lowest BER if there are no non-zero results are used.
Open HTML Report	Opens the HTML report at the path specified in BER Report File.
BER Report File	Defines the path for the HTML report file. See the example report file (see page 157).

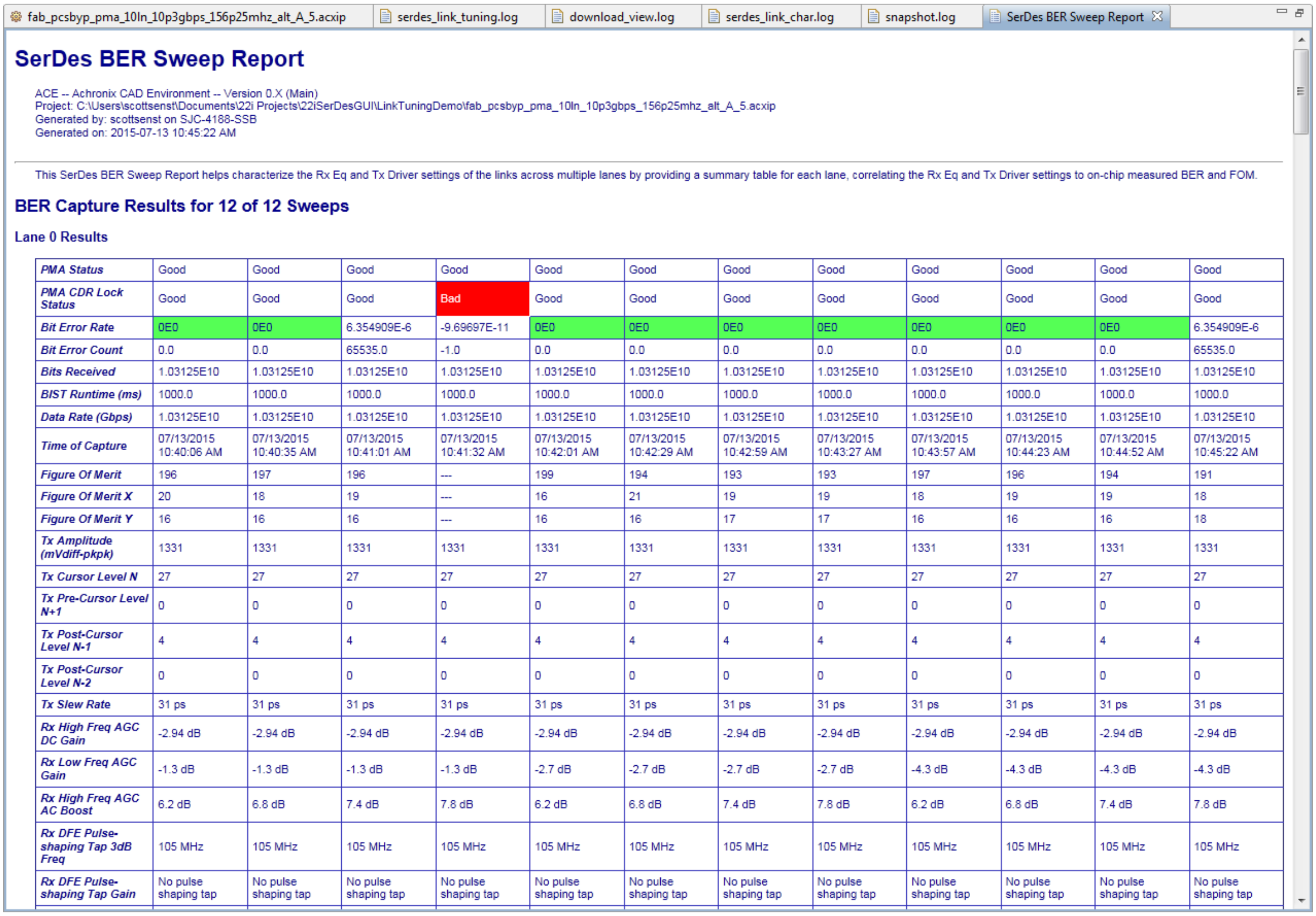
The SerDes PMA BER Capture Sweep Results section displays the current and best results for each enabled SerDes lane. These results are also written to the HTML report file.

Table 56: *SerDes PMA BER Capture Sweep Results*

Item	Editable	Description
SerDes Lanes		Identifies the individual lanes in the channel. The number of lanes corresponds to the Number of Lane setting on the SerDes Overview Page (see page 122).
Enable BER Sweep	✔	Enables which lanes are analyzed the during BER characterization. Restricting characterization to only problematic lanes can greatly reduce runtimes.
PMA Status		Generic status of the PMA. The status is listed as either Good or Bad.
PMA CDR Lock Status		Indicates the receive CDR Lock to Data status. The status is listed as either Good or Bad.
Bit Error Rate		–
Bit Error Count		–
Bits Received		–
BIST Runtime (ms)		The length of time to run the PMA BIST,
Data Rate (Gbps)		Reflects the link data rate. This value is based on the Tx Data Rate setting on the SerDes Overview Page (see page 122) .
Time of Capture		Timestamp for when the sweep results were captured.
Figure of Merit		Returns the figure of merit (FOM) of the signal observed on the receive side of the PMA by using a 4-point eye analysis. This value is captured using the number of iterations specified, and returns the average value across all iterations for all enabled lanes.
Figure of Merit X		Returns the figure of merit based on eye width.

Figure of Merit Y		Returns the figure of merit based on eye height.
Tx Amplitude (mVdiff-pkpk)		Indicates the full-scale maximum swing of the transmit driver
Tx Cursor Level N		Indicates the total number of driver units allocated to the sum of the driver taps.
Tx Pre-Cursor Level N+1		Indicates the total number of driver units allocated to the sum of first pre-cursor tap.
Tx Post-Cursor Level N−1		Indicates the total number of driver units allocated to the sum of the first post-cursor tap.
Tx Post-Cursor Level N−1		Indicates the total number of driver units allocated to the sum of the second post-cursor tap.
Tx Slew Rate		Indicates the transmit driver slew rate.
Rx High Freq AGC DC Gain		Indicates the amount of receive high-frequency AGC DC gain.
Rx Low Freq AGC Gain		Indicates the amount of receive low-frequency AGC gain.
Rx High Freq AGC AC Boost		Indicates the amount of receive high-frequency AGC AC boost.
Rx DFE Pulse-shaping Tap 3dB Freq		Reflects the amount of receive DFE pulse-shaping tap 3dB frequency.
Rx DFE Pulse-shaping Tap Gain		Reflects the amount of receive DFE pulse-shaping tap gain.
Rx DFE N-1 Tap Gain Control (mV)		Indicates the amount of receive DFE tap gain control
Rx DFE N-2 Tap Gain Control (mV)		
Rx DFE N-3 Tap Gain Control (mV)		
Rx DFE N-4 Tap Gain Control (mV)		

The figure below shows a sample sweep report in HTML format.



SerDes BER Sweep Report

ACE -- Achronix CAD Environment -- Version 0.X (Main)
Project: C:\Users\scottsenst\Documents\22i Projects\22iSerDesGUI\LinkTuningDemo\fab_pcsbyp_pma_10ln_10p3gbps_156p25mhz_alt_A_5.acxip
Generated by: scottsenst on SJC-4188-SSB
Generated on: 2015-07-13 10:45:22 AM

This SerDes BER Sweep Report helps characterize the Rx Eq and Tx Driver settings of the links across multiple lanes by providing a summary table for each lane, correlating the Rx Eq and Tx Driver settings to on-chip measured BER and FOM.

BER Capture Results for 12 of 12 Sweeps

Lane 0 Results

PMA Status	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good
PMA CDR Lock Status	Good	Good	Good	Bad	Good	Good	Good	Good	Good	Good	Good	Good
Bit Error Rate	0E0	0E0	6.354909E-6	-9.69697E-11	0E0	0E0	0E0	0E0	0E0	0E0	0E0	6.354909E-6
Bit Error Count	0.0	0.0	65535.0	-1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	65535.0
Bits Received	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10
BIST Runtime (ms)	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0
Data Rate (Gbps)	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10	1.03125E10
Time of Capture	07/13/2015 10:40:06 AM	07/13/2015 10:40:35 AM	07/13/2015 10:41:01 AM	07/13/2015 10:41:32 AM	07/13/2015 10:42:01 AM	07/13/2015 10:42:29 AM	07/13/2015 10:42:59 AM	07/13/2015 10:43:27 AM	07/13/2015 10:43:57 AM	07/13/2015 10:44:23 AM	07/13/2015 10:44:52 AM	07/13/2015 10:45:22 AM
Figure Of Merit	196	197	196	---	199	194	193	193	197	196	194	191
Figure Of Merit X	20	18	19	---	16	21	19	19	18	19	19	18
Figure Of Merit Y	16	16	16	---	16	16	17	17	16	16	16	18
Tx Amplitude (mVdiff-pkpk)	1331	1331	1331	1331	1331	1331	1331	1331	1331	1331	1331	1331
Tx Cursor Level N	27	27	27	27	27	27	27	27	27	27	27	27
Tx Pre-Cursor Level N+1	0	0	0	0	0	0	0	0	0	0	0	0
Tx Post-Cursor Level N-1	4	4	4	4	4	4	4	4	4	4	4	4
Tx Post-Cursor Level N-2	0	0	0	0	0	0	0	0	0	0	0	0
Tx Slew Rate	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps	31 ps
Rx High Freq AGC DC Gain	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB	-2.94 dB
Rx Low Freq AGC Gain	-1.3 dB	-1.3 dB	-1.3 dB	-1.3 dB	-2.7 dB	-2.7 dB	-2.7 dB	-2.7 dB	-4.3 dB	-4.3 dB	-4.3 dB	-4.3 dB
Rx High Freq AGC AC Boost	6.2 dB	6.8 dB	7.4 dB	7.8 dB	6.2 dB	6.8 dB	7.4 dB	7.8 dB	6.2 dB	6.8 dB	7.4 dB	7.8 dB
Rx DFE Pulse-shaping Tap 3dB Freq	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz	105 MHz
Rx DFE Pulse-shaping Tap Gain	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap	No pulse shaping tap

Figure 15: ***Sample Link Characterization Report***

Register Settings - Lane N Pages

The Register Settings pages (there will be one for each enabled lane) allow the user to review and override any/all raw settings in the SerDes configuration registers.


*new_serdes_1.acxip
Speedster22i 12G SerDes
Register Settings - Lane 0
This page allows you to view and configure the raw register settings of a given serdes lane.
AHB Address Filter
Start Address
00000
End Address
1ffff
Function
Any
AHB Register Settings
AHB Address
Override Value
Value
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
AHB Register Details
Generate
<< Back
Next >>
Configuration
File Preview

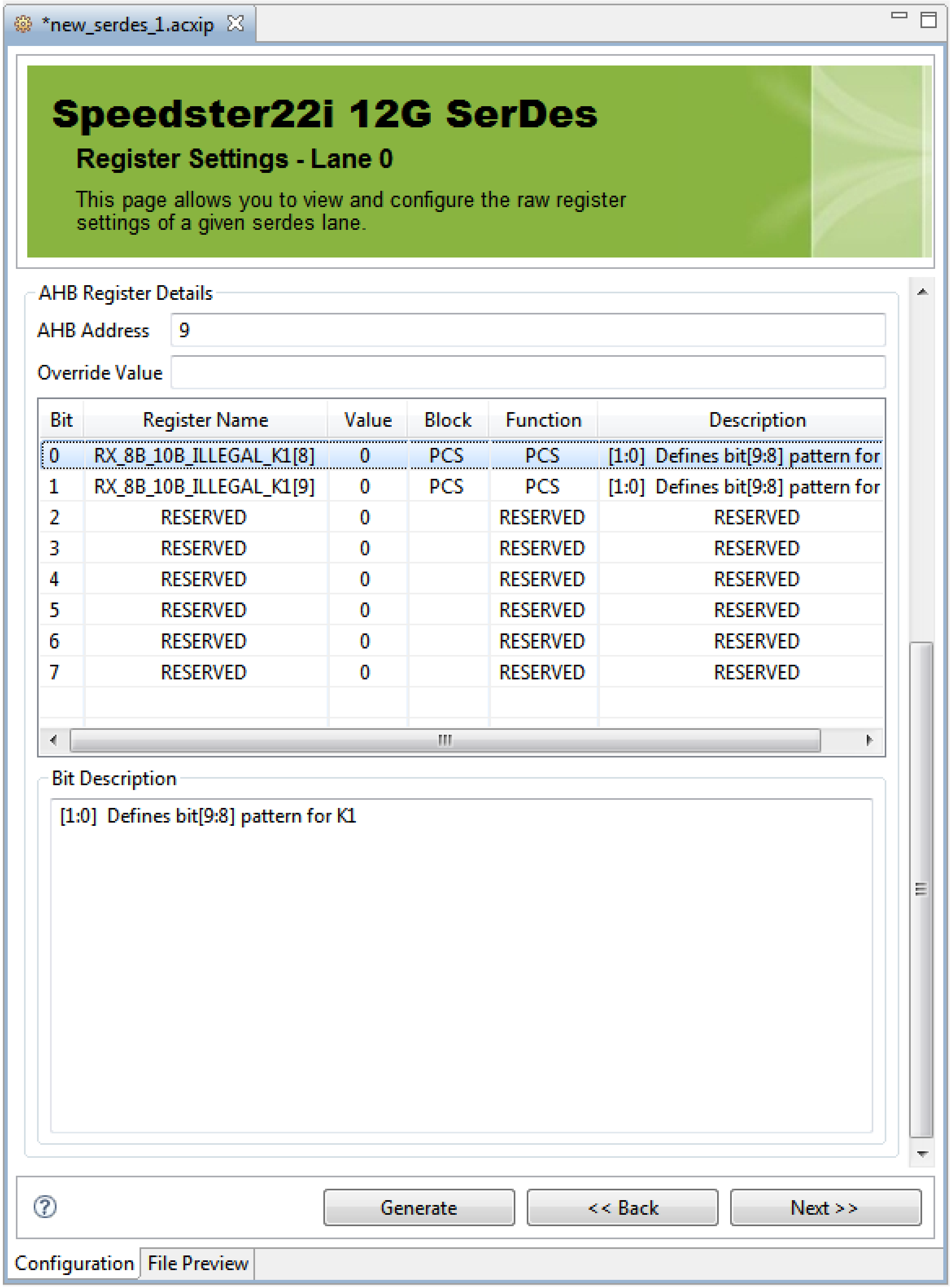

*new_serdes_1.acxip
Speedster22i 12G SerDes
Register Settings - Lane 0
This page allows you to view and configure the raw register settings of a given serdes lane.
AHB Register Details
AHB Address
9
Override Value
Bit
Register Name
Value
Block
Function
Description
0 RX_8B_10B_ILLEGAL_K1[8] 0 PCS PCS [1:0] Defines bit[9:8] pattern for
1 RX_8B_10B_ILLEGAL_K1[9] 0 PCS PCS [1:0] Defines bit[9:8] pattern for
2 RESERVED 0 RESERVED RESERVED
3 RESERVED 0 RESERVED RESERVED
4 RESERVED 0 RESERVED RESERVED
5 RESERVED 0 RESERVED RESERVED
6 RESERVED 0 RESERVED RESERVED
7 RESERVED 0 RESERVED RESERVED
Bit Description
[1:0] Defines bit[9:8] pattern for K1
Generate
<< Back
Next >>
Configuration
File Preview

Table 57: *Register Settings Page Options*

Option	Description
AHB Address Filter	
Start Address	Sets the starting address for filtering the registers displayed in "AHB Register Settings". Settings must be a hexadecimal value.
End Address	Sets the ending address for filtering the registers displayed in "AHB Register Settings". Settings must be a hexadecimal value.
Function	Filters the class of functions displayed in "AHB Register Settings".
AHB Register Settings	
AHB Address	This column displays the register address.
Override Value	This column displays the register override value if set.
Value	This column displays the current register value.
AHB Register Details	
AHB Address	Displays the address of the register selected in "AHB Register Settings".
Override Value	Sets the register value for the register bits displayed under columns "Bit" through "Description". Settings must be a hexadecimal value.
Bit	This column displays the register bit locations.
Register Name	Displays the name for that bit location.
Value	Displays the current bit value.
Block	Displays the block assignment.
Function	Displays the function the bit location belongs to.
Description	Displays a description of the bit location.
Bit Description	Displays a description of the bit location selected in the table. Identical to the "Description" column, but provides additional reading area for long descriptions.

Speedster22i Shift Register Configuration Editor

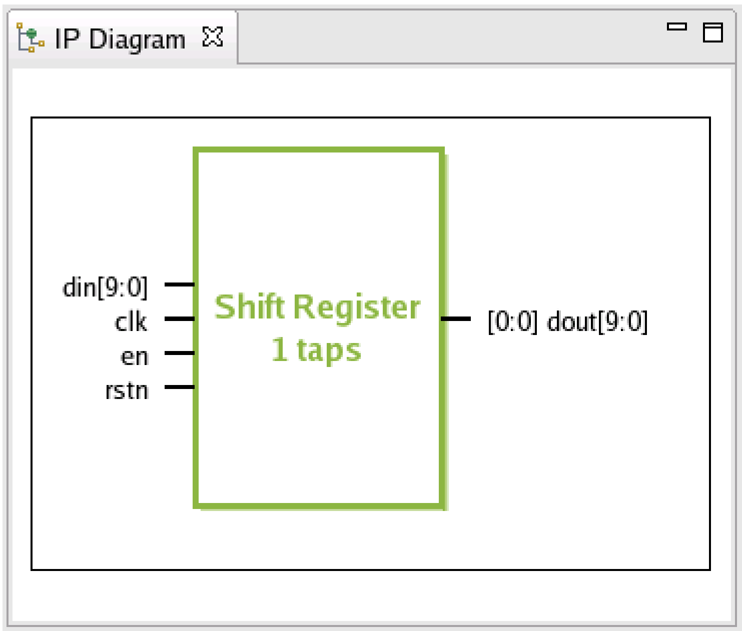
The Shift Register Configuration Editor provides a simplified graphical wizard for creating a shift register configuration file (.acxip). This editor allows the user to generate the required configuration files for design with the embedded Shift Registers. See Creating an IP Configuration (see page 387).

By default, the Shift Register Configuration Editor is included in the IP Configuration perspective (**Window -> Open Perspective -> IP Configuration**).

Once the user has configured the shift register to meet their requirements, and the Shift Register Configuration Editor has determined that there are no errors in the configuration, the user may choose to generate their IP design files (see Generating the IP Design Files (see page 389)).

IP Diagram

The IP Diagram View for the Shift Register shows live information about the current configuration in the Editor, including which inputs and outputs are currently active.



1. Overview Page (see page 161)
2. Tap Settings Page (see page 162)

Shift Register Overview Page

The Overview page contains the top-level, global properties that govern the structure and base configuration of the Shift Register wrapper.

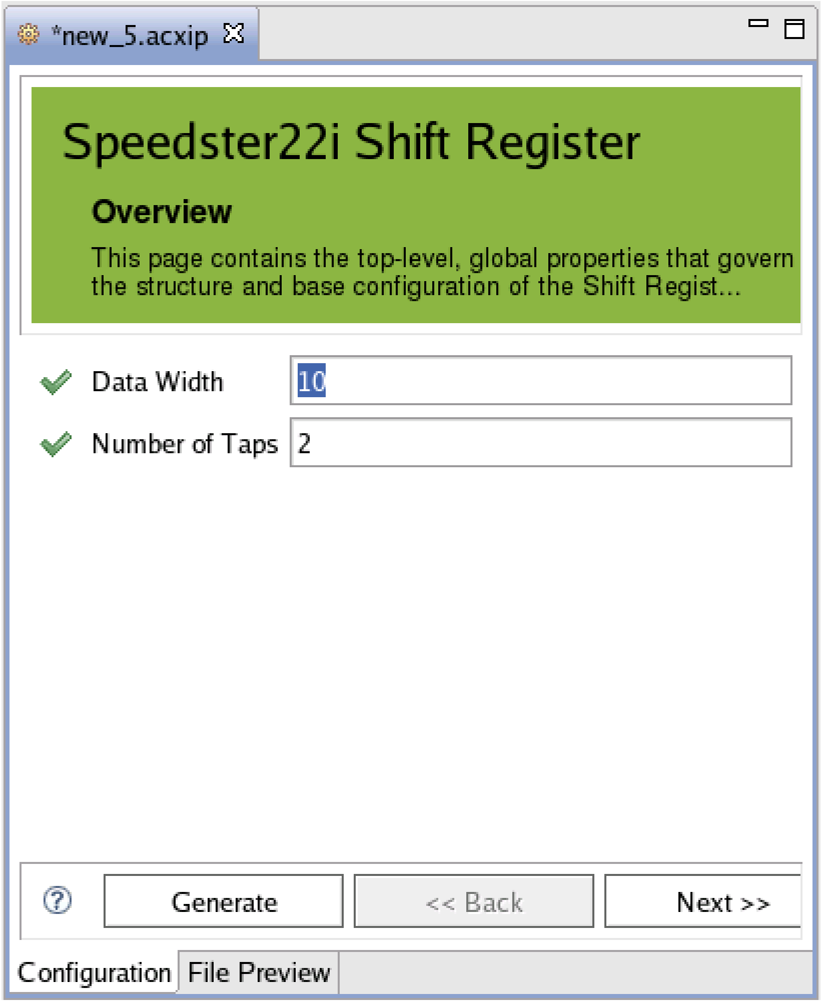


Table 58: *Shift Register Editor Overview Page Options*

Option	Editable	Description
Data Width	Y	Data width of read and write ports; allowed range is 1 - 65536.
Number of Taps	Y	Number of taps including the final Tap. The leftmost entry corresponds to the first tap-off after input; allowed range is 1 - 256.

Tap Settings Page

The Settings page contains the settings for each tap.

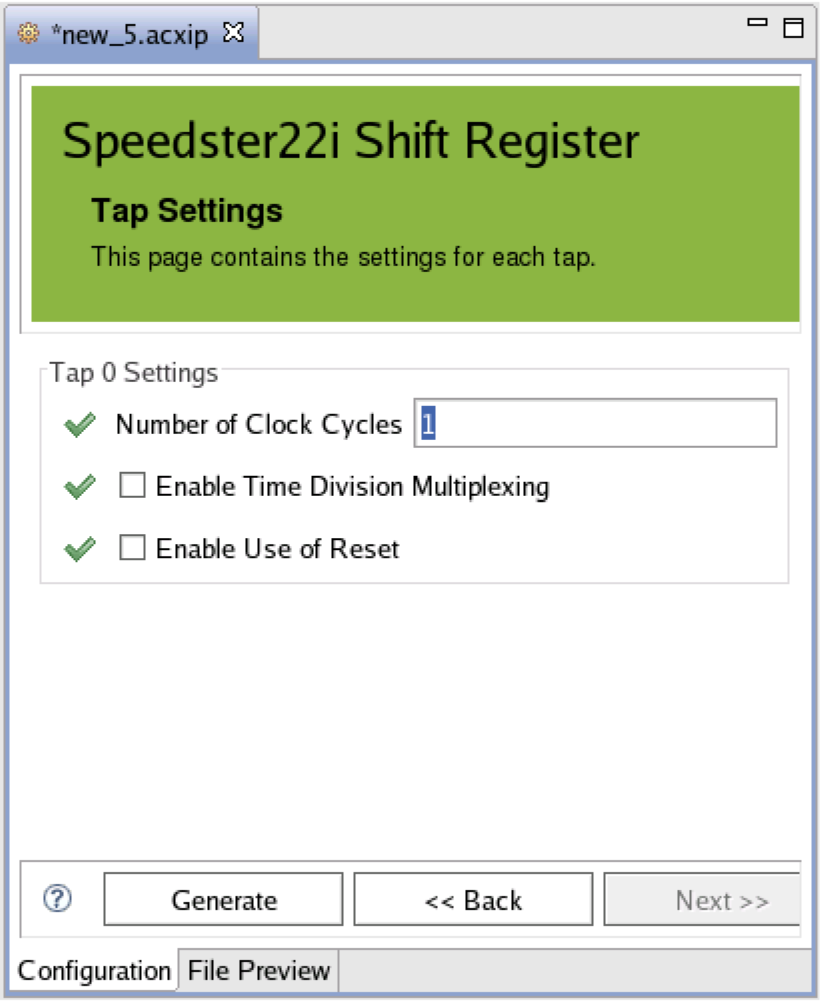


Table 59: *Shift Register Editor Settings Page Options*

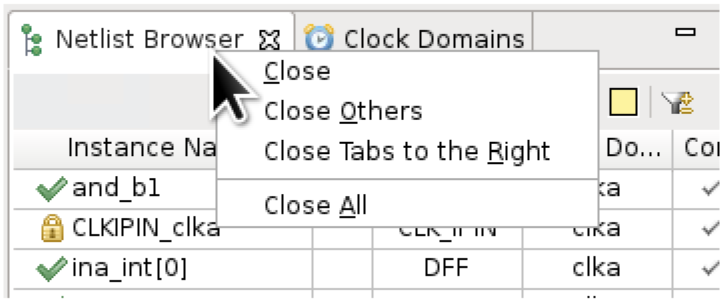
Option	Editable	Description
Number of Clock Cycles	Y	Absolute number of clock cycles to shift from the input; allowed range is 1 - 65536.
Enable Time Division Multiplexing	Y	Time-Division-Multiplex `i`th and `i+1`th tap-off.
Enable Use of Reset	Y	If turned off, the `rstn` input is ignored.

Views

Views support Editors (see page 25) and provide alternative presentations as well as ways to navigate the information in the Workbench (see page 23). For example, the Projects View (see page 238) displays Projects (see page 312), Implementations (see page 312), 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's tab to display the menu.

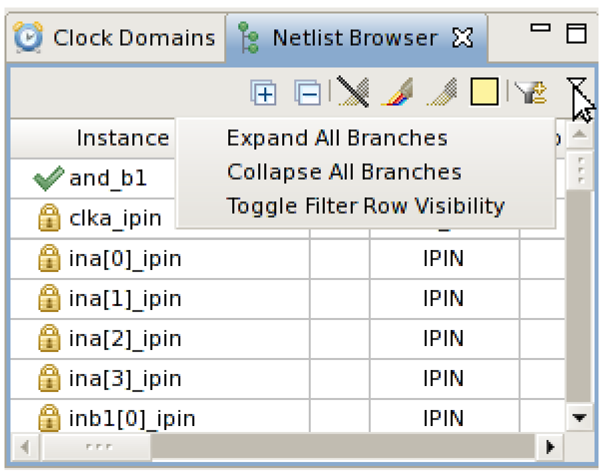
Figure 16: ***Example of a view's right-click context menu***



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 small down arrow icon (▽) will appear at the far right of the view's toolbar. To open the menu for a view, click the (▽) icon.

Figure 17: ***Example of a view's toolbar, including an opened view menu***



Views are typically grouped by shared context into Perspectives (see page 23). 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 23) window. See Working with Views and Editors (see page 345) for more information.

The views contained by the Project Perspective are the Projects View (see page 238), Flow View (see page 187), Options View (see page 215), and Tcl Console View (see page 258). The Multiprocess View (see page 204) is also part of this perspective, but is hidden by default.

The views within the Floorplanner Perspective are the Search View (see page 245), Selection View (see page 249), Critical Paths View (see page 174), Critical Path Diagram View (see page 170), Floorplanner View (see page 178), Package View (see page 226), IO Assignment View (see page 191), Clock Regions View (see page 167), Clock Domains View (see page 165), Placement Regions View (see page 234), Partitions View (see page 231), Netlist Browser View (see page 211), Properties View (see page 242), and Tcl Console View (see page 258).

The views contained within the IP Configuration Perspective are the Outline View (see page 225), IP Libraries View (see page 197), IP Diagram View (see page 196), IP Problems View (see page 198), and Tcl Console View (see page 258).

The views of the Programming and Debug Perspective are the Snapshot Debugger View (see page 253), Download View (see page 176), JTAG Browser View (see page 199), JTAG Diagram View (see page 203), and Tcl Console View (see page 258).

The views of the HW Demo Perspective are the HW Demo View (see page 189), Snapshot Debugger View (see page 253), and Tcl Console View (see page 258).

Clock Domains View

The Clock Domains view will provide 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 211), filters are available for each column of the table, so that in cases where there are many clock domains in a design, users may limit the visible content of the table to just those clock domains meeting their 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 23). To add it to other perspectives, select **Window** → **Show View...** → **Other...** → **Achronix** → **Clock Domains**.

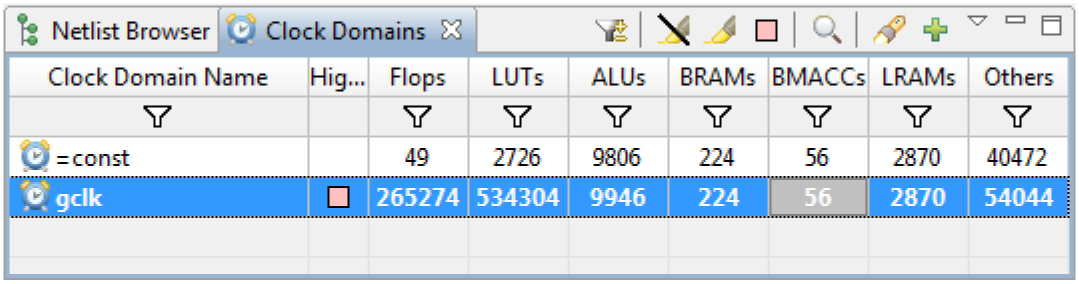


Figure 18: *Screenshot of the Clock Domains view*

> The various Achronix target devices contain different mixes of the possible resource types. Accordingly, the resource type columns in this view (such as Flops, BRAMs, ALUs, etc.) are dynamic, and will change to match the target device after the Prepare flow step has been run. The screenshots and example descriptions in this document section may not exactly reflect the user's current target device.

Table 60: *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 will show 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 will display no color square.
Resource	A different column is provided for each *resource* type within the target device. Each table cell in that column will show the sum count of all contained *resource* instances within the row's named clock domain.

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 61: *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 249)).
	Choose Highlight Color	Y		Determines which color will be 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 396) .)
	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 178) and Package View (see page 226) 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 245) is populated with the results.
	Toggle Filter Row Visibility	Y		Changes whether the filter row (of filter icons) is visible or not. Note that this does not alter whether filters are active, it only changes the visibility of the row of filter icons.

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 by placing the mouse cursor over the boundary between columns (the mouse cursor changes to indicate resizing is possible). Next, 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 by left-clicking and holding on any column name, then dragging left or right to move the column between any other pair of columns, releasing 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 will occur 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 (glob string matching by default, but regular expression matching is also available, following Java rules, which are extremely similar to Perl rules). Columns with check marks can be filtered by Boolean value. Columns containing numbers can be filtered by numerical value. Columns which cannot be filtered (such as, the Highlight Color column) lack a filter icon in the filter row.

To add a filter to a column, the Filter Row must first be visible (select the **Toggle Filter Row Visibility** action to show the row if necessary). Then simply left-click the mouse on the filter icon ( ) for the desired column, which causes a data-appropriate filter dialog to appear. Next, fill in the desired filter values and click **Apply** to apply the filter to the clock domains 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 changge to the active filter icon ( ).

To edit (or clear) an existing filter, simply left-click the mouse on the active filter icon ( ), which again causes the data-appropriate filter dialog to appear, pre-populated with the current column filter setting. Change the filter value and click **Apply** again to edit the filter, click **Cancel** to leave the filter unchanged, or 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 23). 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 258), 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.

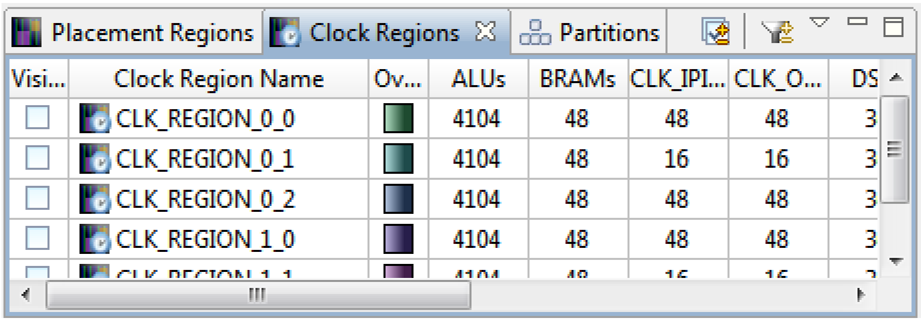
Any clock domain in the table may be dragged to the Placement Regions view (see page 234) or the Floorplanner View (see page 178) (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 438). Dragging a clock domain is the equivalent of dragging all 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 will not be assigned to placement regions.

Clock Regions View

The Clock Regions view provides a tabular representation of the site type content of each of the Clock Regions (see page 328) in the currently selected Target Device, and allows the user to toggle the visibility of the overlay within the Floorplanner View (see page 178) for each Clock Region. The view's table will remain 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 23). To add the view to the current perspective, select **Window -> Show View -> Other... -> Achronix -> Clock Regions**.

Figure 19: ***Screenshot of Clock Regions View***



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 will not match all target devices.

Table 62: *Clock Regions Table Columns*

Column	Editable	Description
Visibility	Y	When checked, the clock region overlay is painted in the Floorplanner View (see page 178), 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.
Resource		The number of *resource* sites contained in this clock region

Table 63: *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 the user to change the translucent overlay color which will be 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's 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 will pseudo-randomly pick another color. Each time this action is chosen, another color will be picked.
					Pans and zooms the Floorplanner View to show the location of the selected clock region. (Note that if the clock region's visibility column

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Zoom To		Y		checkbox is disabled, the clock region overlay will not be painted and thus will not be visible.)
	Reset All Overlay Colors			Y	Pseudo-randomly reassigns new overlay colors for all clock regions.
	Show / Hide All Clock Regions	Y	Y	Y	This will toggle the visibility checkboxes for all clock domains in the table, causing them 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. Note that this 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 178). By default, no clock region overlays are painted in the Floorplanner View. Users must choose to enable the clock region overlays they wish to have displayed. Users may optionally alter the overlay color for each/all clock regions, but these color choices are not persisted between ACE sessions.

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

The following are ways to alter the presentation of Clock Region data in the Floorplanner View (see page 178):

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 will be painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view will be redrawn with the chosen clock region overlay no longer painted.

Enable/Disable painting of all clock regions within the Target Device:

When the user chooses the **Show/Hide All Clock Regions** action, the visibility of all clock regions will be 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 by right-clicking the mouse pointer anywhere on the row of the desired clock region, then selecting **Choose Overlay Color** from the popup context menu. The user will then be able to use the Color Dialog to choose the desired color for the clock region. Note that this is a temporary color change - colors will be reverted to automatically chosen defaults if the user changes the active design, the active implementation, the target device, or closes ACE.

ACE will automatically pick a different overlay color for an individual clock region if the user chooses **Reset Overlay Color** from the right-click popup content menu.

Temporarily alter the overlay rendering color for all clock regions:

ACE will automatically pick different overlay colors for all clock regions if the user chooses the **Reset All Overlay Colors** action from the Clock Regions View's 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 by placing the mouse cursor over the boundary between columns - at this point the mouse cursor should change to indicate resizing is possible. Next, 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 by left-clicking and holding on any column name, then dragging left or right to move the column between any other pair of columns, and release the left mouse button to insert the column header at the new location. While dragging, you will see the dragged column header alongside the mouse cursor, and there will be a thick column header separator showing where the column insertion will occur 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 will be retained; non-matching values will be excluded from the table.

Columns containing text can be filtered by string value (glob string matching by default, but Regular Expression matching is also available, following Java rules, 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 (like the Overlay Color column) will lack a filter icon in the filter row.

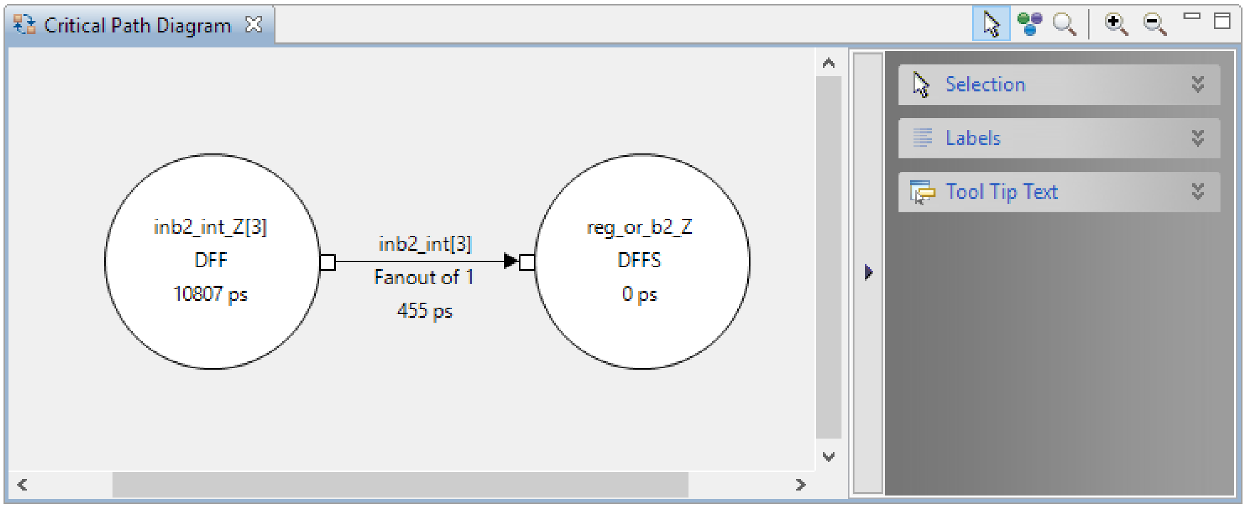
To add a filter to a column, the Filter Row must first be visible. (Select the **Toggle Filter Row Visibility** action to show the row if necessary.) Then simply left-click the mouse on the filter icon ( ) for the desired column, which causes a data-appropriate filter dialog to appear. Next, fill in the desired filter values and press **Apply** to apply the filter to the clock regions in the table. All values matching that filter will be retained, and all other values will be excluded. Additionally, the background color of the filtered column will change to a bright yellow to indicate the filter is active, and the filter icon at the head of the column will also change to the active filter icon ( ).

To edit (or clear) an existing filter, simply left-click the mouse on the active filter icon ( ), which again causes the data-appropriate filter dialog to appear, this time pre-populated with the current column filter setting. Change the filter value and press **Apply** again to edit the filter, press **Cancel** to leave the filter unchanged, or press **Clear** to remove the filter from the column. If the filter is cleared, the background color of the column will return to the default background color, and the filter icon will also change to the inactive version ( ).

Critical Path Diagram View

The Critical Path Diagram view will provide a graphical representation of a single critical path. Selecting a row in the Critical Paths View (see page 174)'s table will cause the Critical Path Diagram view's diagram to update so that it contains a graphical representation of the selected critical path. The graphical representations will consist of circular nodes (representing instances) connected by arrows (representing one or more nets). Similar to the Floorplanner View (see page 178) and Package View (see page 226), 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 290). For details on usage of the diagram view, please see Using Critical Path Diagrams (see page 407) or Analyzing Critical Paths (see page 404).

By default, the Critical Path Diagram view is included in the Floorplanner perspective (see page 23). To add it to the current perspective, select **Window** → **Show View...** → **Critical Path Diagram**.



When no critical path is selected in the Critical Paths View (see page 174), the diagram will display a warning.

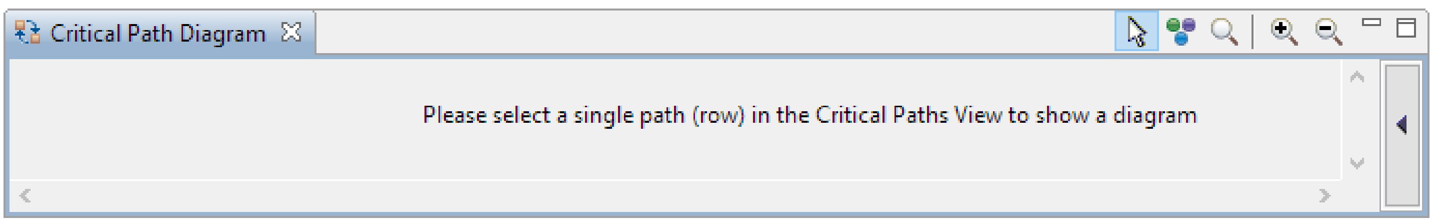


Table 64: *Critical Path Diagram View Toolbar Buttons*

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 the user drags 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 the user drags 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%.

The user may also right-click on an instance or net within the diagram to display a context menu of additional actions.

Table 65: *Critical Path Diagram View Context Menu Actions*

Icon	Action	Description
	Add to Selection	The instance or net under the mouse will be added to the ACE selection set (and be painted in the selection color).
	Remove from Selection	The instance or net under the mouse will be removed from the ACE selection set if currently selected (and will thus no longer be 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 will be 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	Attempts to open a text editor to the file and line number relevant to the instance or net under the mouse. **Caution!** This is Early Access functionality; this may not always open the text editor to the expected location.
	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).
	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 249)) to be unplaced at once (this is much more efficient than unplacing multiple instances individually).

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 66: *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 may be necessary to **Zoom In** to provide sufficient area for all the desired text to be displayed.

Table 67: *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 68: *Tooltip Options*

Option	Description
None	No tooltips will be displayed for nodes or arrows.
Instance Names	Displays the instance name each graph node represents.

Option	Description
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 170)) displays critical path details, manages selection of objects on critical paths, and highlights critical paths in the Floorplanner View (see page 178).

Clicking on a row in the table enables the toolbar buttons for analyzing the associated critical path, and causes a graphical diagram of the associated critical path to be displayed in the Critical Path Diagram View (see page 170). Clicking on a column header sorts the table according to that column's data.

By default, the Critical Paths view is included in the Floorplanner perspective (see page 23). To add it to the current perspective, select **Window** → **Show View...** → **Critical Paths**.

Table 69: *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	Prints 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, allowing the user to save 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 will be 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 will take precedence over hold violations as the branch node, regardless of relative slack values.

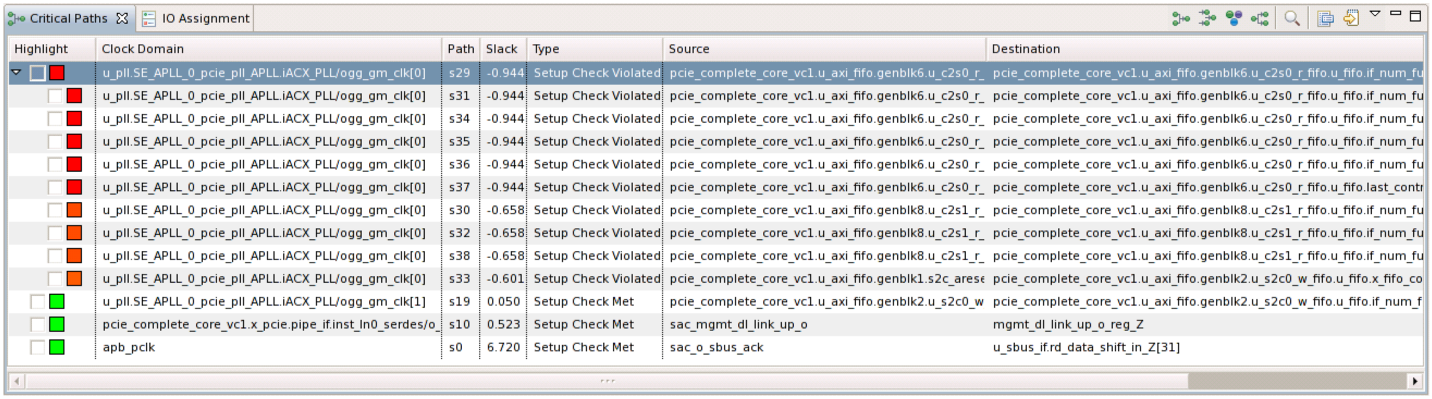


Figure 20: *Screenshot of the Critical Path View*

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.

Default sort order, from most critical to least critical, of the critical paths:

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 70: *Critical Path View Table Columns*

Column	Description
Highlight	Allows the user to highlight the path in the Floorplanner view, using the checkbox. Also allows the user to configure the highlight color of the path by clicking on 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 322)).
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 will range 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 will always be green, and will not vary by reported slack value.

The View's local pull-down menu (found to the right of the View's 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 71: *Critical Paths View Drop-down Menu Actions*

Action	Description
Show Clock Paths	If checked, highlighted critical paths in the Floorplanner View will show the clock routing segments as part of the critical path. If unchecked, only the data portion of the critical path will be displayed.
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.
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 23). To access the Download view, select **Window -> Show View... -> Others -> Download View**.

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

> While using the Download view, it is strongly recommended that the Tcl Console View (see page 258) be kept visible to display any status or error messages returned from the external `acx_stapl_player` process.

> **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 288), which is easily accessible by pressing the **Configure JTAG Interface** ( ) button in the Download view. See Configuring the JTAG Connection (see page 411) for more details.

See also: Playing a STAPL File (Programming a Device) (see page 432).

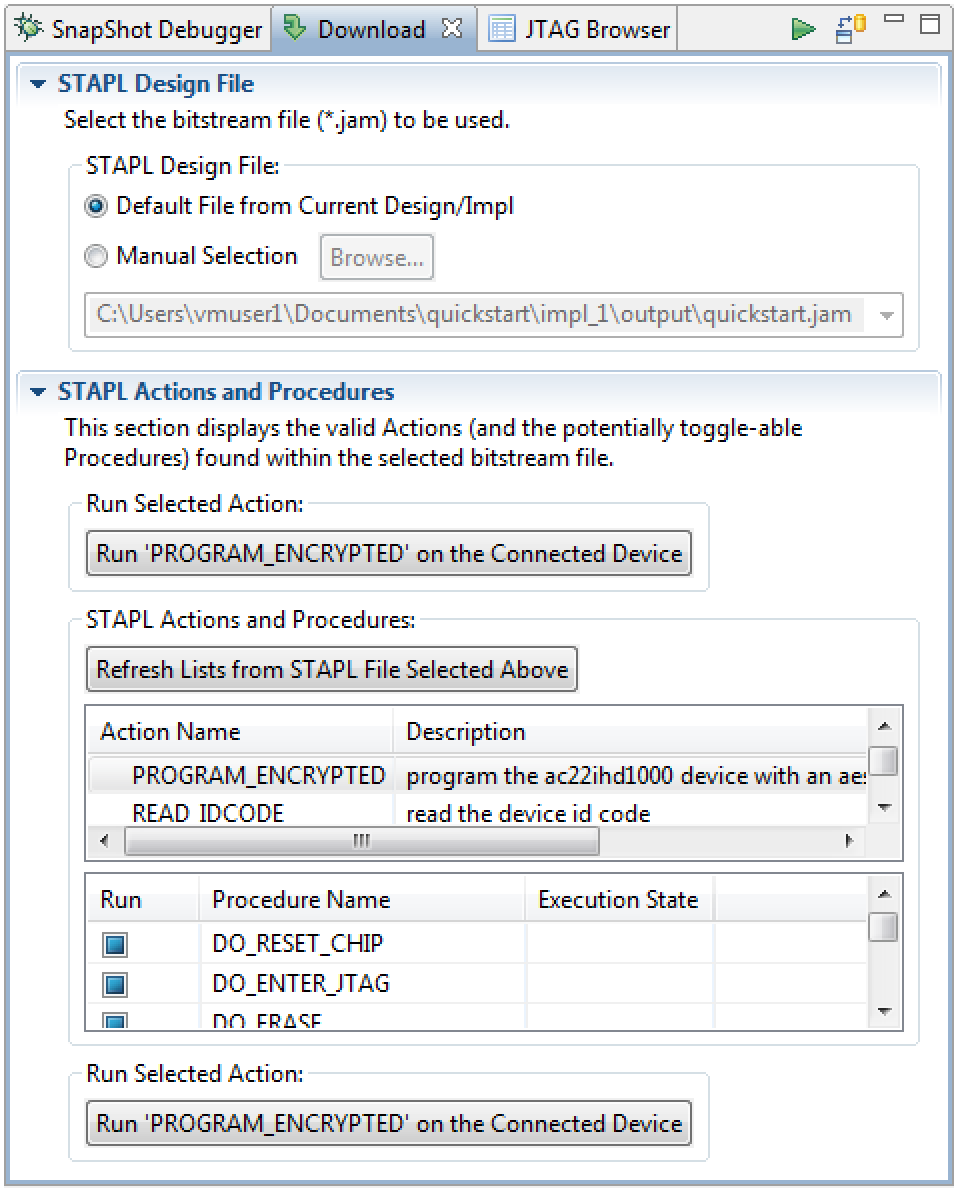
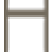
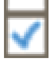


Table 72: *Download View Options*

Option	Description
STAPL design file	
Default File from Current Design / Impl	The STAPL design file will correspond to the bitstream file of the currently active implementation (`design_name.jam`), as generated during the **Generate Bitstream** Flow Step (see page 317).
Manual Selection and Browse	Allow the user to choose / enter any STAPL design file from the file system. The textfield showing the filename is also a drop-down 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 (will always be run; cannot be deselected)  = Deselected (will not be run; may be toggled)  = Selected (will be 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*' on the Connected Device	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 the user to visualize the device, place and route data, critical paths, and the current selection set. The view allows the user to zoom out to see a general overview of the user's design mapped onto the device, or the user may zoom in to see specific details.

Clicking on the tall narrow arrow button on the far right of the Floorplanner view shows or hides the Fly-Out Palette (see page 181) of display options.

By default, the Floorplanner view is included in the Floorplanner perspective (see page 23). To add it to the current perspective, select **Window → Show View → Other... → Floorplanner**.

See also: Viewing the Floorplanner (see page 392), Pre-placing a design (see page 399), Floorplanner View Colors and Layers Preference Page (see page 292), and Floorplanner View Optimizations Preference Page (see page 296).

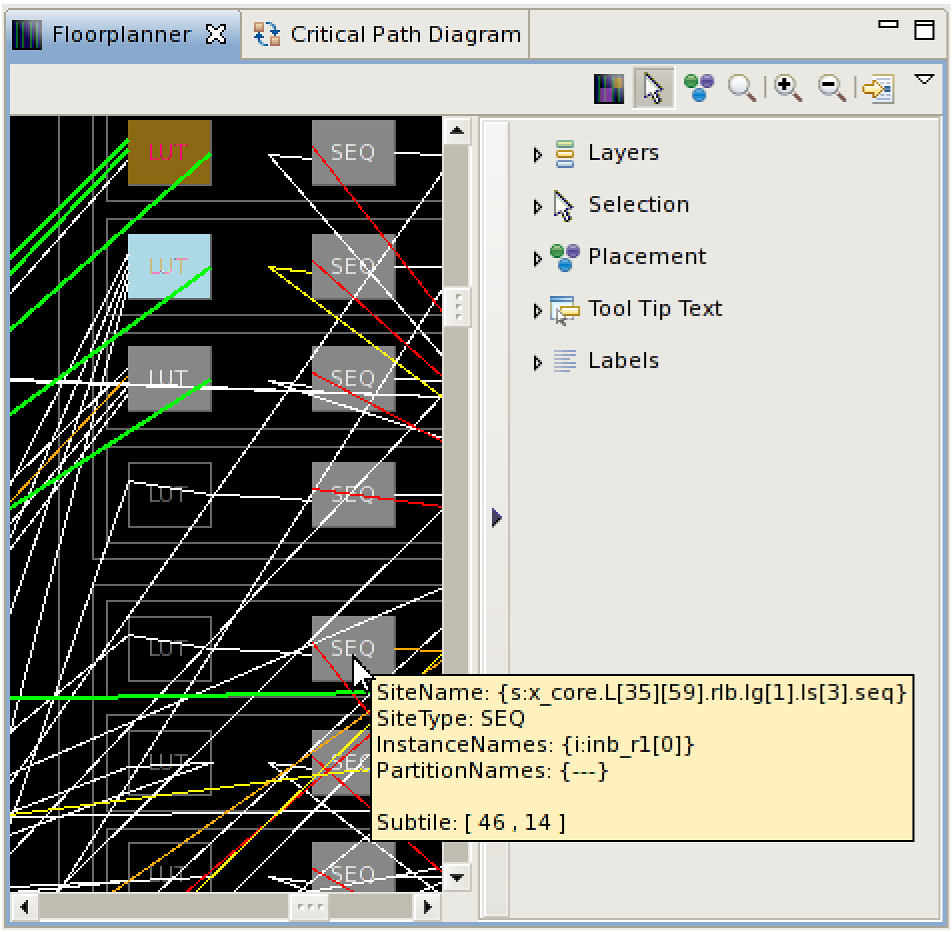
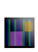


Figure 21: ***Screenshot of Floorplanner View, Including Expanded Fly-Out Palette***

Table 73: ***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 435). When the Placement Region Tool is active, the user may use the mouse to create new placement regions (see page 436), move (see page 438) or resize (see page 437) existing placement regions, and/or assign objects to placement region constraints (see page 438).
	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.

Icon	Action	Description
	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.
	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 281) allowing the user to save the current placement to a pre-placement constraints file (`.pdc`).

The user may also right-click on an Instance or Net within the Floorplanner to display a context menu of additional actions.

Table 74: *Floorplanner View Context Menu Actions*

Icon	Action	Description
	Add to Selection	The Instance or Net under the mouse will be added to the ACE selection set (and be painted in the Selection color).
	Remove from Selection	The Instance or Net under the mouse will be removed from the ACE selection set if currently Selected (and will thus no longer be 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 will be 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	Attempts to open a text editor to the file and line number relevant to the Instance or Net under the mouse. ⚠ This is Early Access functionality; this may not always open the text editor to the expected location.

	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).
	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 249)) to be unplaced at once. (This is much more efficient than unplacing multiple instances individually.)

Panning and Zooming

The Floorplanner view allows the user to zoom in and out, to see more or less details respectively. There are several ways the user may change the zoom level: 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 Floorplanner In and Out (see page 393) 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 will 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 will require 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's drag-and-drop interactions. See the task Floorplanner Panning (see page 393) for complete details.

> When painting objects in the Floorplanner, when the view is zoomed out, some objects will become too small to be rendered with any detail. These objects will be 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 will not be painted at all, even if layer settings would otherwise allow them to be visible. Selected sites will always be painted, with a minimum size of a single pixel.
>
> When a single pixel represents multiple objects, as will happen when zoomed all the way out, ACE will paint only the most critical or most important object state at that pixel location, so the single pixel will be the most critical or most important color. The relative priorities of the states are described in Instance States (see page 328).

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 a user to filter the view so it contains a desired subset of all the available information.

Table 75: *Layer Options*

Option	Default	Description
Instances	Enabled	This layer shows all placed instances.
Selected Instance Flylines	Disabled	This layer shows flylines representing the net connections of selected instances (instances in the ACE selection set). **Note** 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.
Clock Nets	Enabled	This layer shows all clock nets.
Non-clock Nets	Enabled	This layer shows all non-clock nets.
Routing Status		
Open Connections	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**. **Performance Note** 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.
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.

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 will be 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) will be rendered in the center of the placed instance instead of on a specific pin. (Open Connections and Open Pins will not, of course, be rendered for unplaced instances.)

Be aware that in a placed design that has not yet been routed, all nets will be considered Open Connections. Enabling **Open Pins** can make it much easier to find unrouted portions of a mostly routed design when zoomed out. But users should be aware this may 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 will display 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 249)) 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 will still be painted in the Selected Instances color (by default a bright green). Details of this behavior may be tweaked on the Floorplanner View Colors and Layers Preference Page (see page 292).

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 167); the visibility of individual Placement Regions is controlled from the Placement Regions view (see page 234); and the visibility of individual Critical Paths is controlled from the Critical Paths view (see page 174).

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 will only be visible in the Floorplanner if the appropriate Layer is enabled, and the highlight color will only be 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 328), and can be partially reconfigured in the Floorplanner View Colors and Layers Preference Page (see page 292). Additional info regarding highlighting may be found at Highlighting Objects in the Floorplanner View (see page 396).)

Selection

The Selection Options control the selection of objects with the mouse in the Floorplanner view. Selected objects will be added to the ACE Selection Set, and displayed appropriately in the Selection View (see page 249).

Table 76: *Selection Options*

Option	Default	Description
Instances	Enabled	This option enables visible instances to be selected. If not checked, instances in the selection region are not added to the ACE selection set.
Nets	Enabled	This option enables visible nets to be selected. If not checked, nets in the selection region are not added to the ACE selection set.
Pins	Disabled	This option 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	This option enables visible paths to be selected. If not checked, paths in the selection region are not added to the ACE selection set.

Option		Default	Description
Sites		Disabled	This option 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	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.
	Remove Placement	Disabled	This radio button controls the action applied to enabled objects in the selection region. This setting causes the placed instances to be un-placed.
	Fix Placement	Disabled	This radio button controls the action applied to enabled objects in the selection region. This setting causes the soft-placed instances to attempt to have fixed placement at the same site.
	Un-fix Placement	Disabled	This radio button controls the action applied to enabled objects in the selection region. This setting 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 will not be possible to perform selection actions upon instances in the Floorplanner View using the mouse.

For example, this will allow users to perform 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 77: *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 will only succeed 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 may be changed by the placer.

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 78: *Tooltip Options*

Option	Default	Description
Allow Tooltips	Enabled	Allows users to enable/disable 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.
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	Enabled	Includes the subtile coordinates under the current mouse position in the tooltip text. **Note** Subtile coordinates may be used with placement region commands on the Tcl command line.
Partition Names	Enabled	Includes all partition names under the current mouse position in the tooltip text.
IO Package /Ball Names	Enabled	Includes the I/O package/ball names of I/O buffer sites under the current mouse position in the tooltip text.
IO Use	Enabled	Includes the I/O use label of I/O buffer sites under the current mouse position in the tooltip text.

Option	Default	Description
IO LVDS Polarity	Enabled	Includes the I/O LVDS polarity of I/O buffer sites under the current mouse position in the tooltip text.
IO Clock and Reset Capability	Enabled	Includes the clock and reset capability of the I/O buffer sites under the current mouse position in the tooltip text. D = data only; RD = reset and data capable; CRD = clock, reset, and data capable.

Tooltips are filtered by Layers Visibility

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

Label

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

Table 79: *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.
IO Package/Ball Names	Disabled	This option displays the I/O package/ball names on I/O buffer sites.
IO Use	Disabled	This option displays the I/O use label on I/O buffer sites.
IO LVDS Polarity	Disabled	This option displays the I/O LVDS polarity on I/O buffer sites.
IO Clock and Reset Capability	Disabled	This option displays the clock and reset capability on the I/O buffer sites. D = data only; RD = reset and data capable; CRD = clock, reset, and data capable.

Flow View

The Flow view provides a hierarchical view of Flow Steps (see page 317) that can be performed on the Active Project and Implementation (see page 316). From here, flow steps can be run and Flow Status (see page 321) viewed. Flow steps are not able to run unless an active implementation is selected in the Projects View (see page 238). When running flow steps, the implementation options (see page 215) 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 23). To add it to the current perspective, click **Window** → **Show View** → **Other...** → **Achronix** → **Flow**.

For more details, see the Flow (see page 316) concept and the tasks for Running the Flow (see page 361).

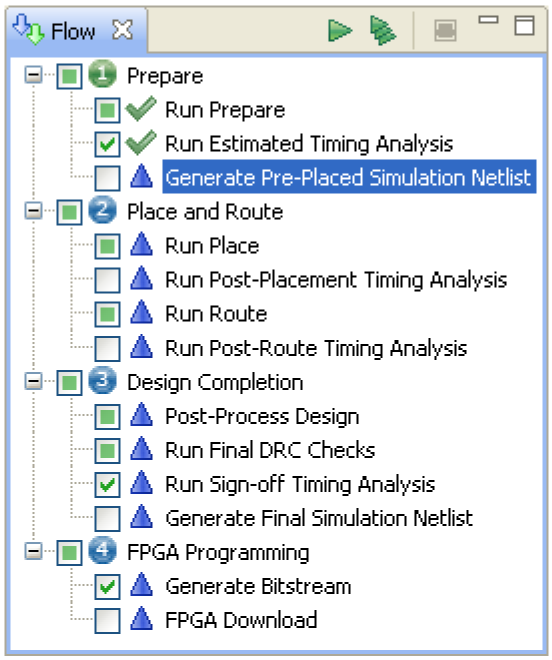


Table 80: *Flow View Icons*

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 378).

Table 81: *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, then this action behaves identically to **Run Flow**.
	Show Multiprocess View	Y		Launches the Multiprocess View (see page 204), which allows the user to manage multiple runs of the flow in parallel.
	Stop flow	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. **Note** Some flow steps, such as FPGA Download, are currently unable to be interrupted while running.
	Run Selected Flow Step		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 will be skipped. **Note** Double-clicking on a flow step is equivalent to this action.
	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"		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). **Caution!** This action is only relevant when Using Incremental Compilation (Partitions) (see page 443). If Incremental Compilation is disabled, this action behaves identically to **Re-Run Flow**.

Icon	Action	Toolbar Button	Context Menu	Description
	Clear Flow		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. **Note** This 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.
	Create Flow Step		Y	Displays an interactive dialog for creating a user defined flow step (the dialog includes a prompt for which single TCL command should be invoked for this step) at the selected location within the flow. **Caution!** Creation of user-defined flow steps is only recommended for advanced users.
	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 can not be used to remove 'reserved' steps.

The current Flow Mode setting impacts which Flow Steps will be executed.

The implementation option for Flow Mode (see page 321) affects which flow steps will be 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

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 288). See Configuring the JTAG Connection (see page 411) for more details.

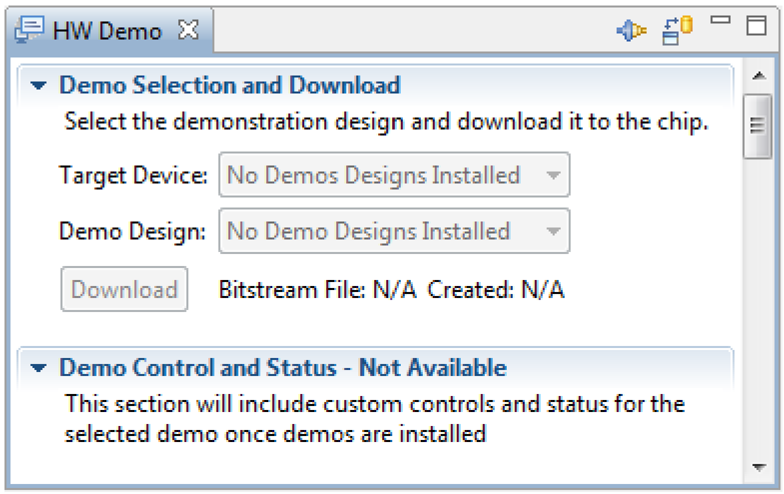
The DCC connection must be configured before using the HW Demos!

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 288). See Configuring the DCC Connection (see page 410) 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 23). 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 will be no demo designs available. In that case, the view will display minimal information:



After demos are installed as ACE overlays, from this view the user will be able to view the status of various board components updating in real-time as the demonstration design is running on the connected board. For example, in a basic fabric demonstration, when the user changes an associated DIP switch 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 will come 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 441).

Table 82: *HW Demo View Toolbar Buttons*

Icon	Action	Description
	Configure DCC Interface	Opens the preferences dialog to the Configure DCC Connection Preference Page (see page 288).
	Configure JTAG Interface	Opens the preferences dialog to the Configure JTAG Connection Preference Page (see page 288).

Table 83: ***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 LED's from the attached board. When an LED changes state on the board, it is reflected in the view's LED display. Clicking on an individual LED in the view will cause 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 on an individual switch in the view will ***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 that the Board Status section may not be present in all HW Demo (and reference) designs.

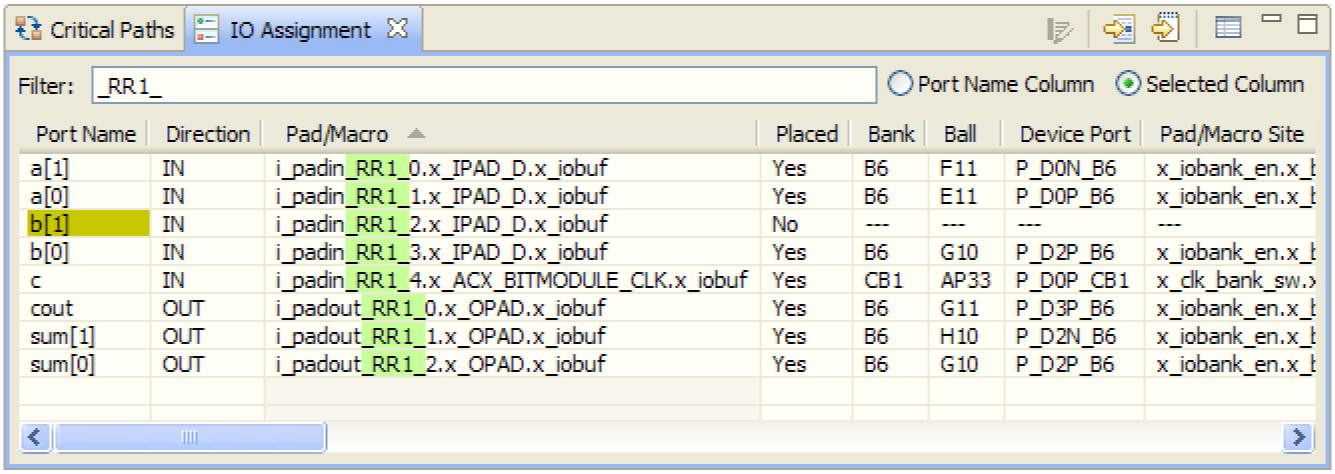
IO Assignment View

The IO Assignment View is only applicable for Speedster FPGA devices

Users should ignore this View 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 will remain empty until the currently active Implementation has been prepared (had the **Run Prepare** flow step completed).

By default, the IO Assignment view is included in the Floorplanner Perspective (see page). To add the view to the current perspective, select **Window -> Show View -> Other... -> IO Assignment**. See also: Managing I/Os (see page 501).



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

In the figure above, there's a filter of "_RR1_" on 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's 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 presentation of the data in the IO Assignment table may be altered in several ways:

- Column resizing: The width of a column can be changed by placing the mouse cursor over the boundary between columns - at this point the mouse cursor should change to indicate resizing is possible. Next, 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 by left-clicking and holding on any column name, then dragging left or right to move the column between any other pair of columns, and release the left mouse button to insert the column header at the new location. While dragging, you will see the dragged column header alongside the mouse cursor, and there will be 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 will occur if the mouse is released at the present cursor location.
- Sorting by column: The data rows can be sorted by the contents of any column. Simply left-click the column header, and the rows will be sorted by the data in that column. Left-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 will not be displayed if their data for the selected column does not match the current filter value. All displayed rows will, in the selected column, contain text that matches the applied filter. The portion of the text that matches the filter will be highlighted (in green). By default, the column being filtered is the **Port Name** column.
  To select an alternate column for the filter, first select the **Selected Column** radio button, then left-click the corresponding column header. The selected column will be indicated in a platform-specific manner. (Windows: the column header will have a small arrow indicating the sort direction of the selected column, and any visible empty rows will have the selected column in a slightly different color than the non-selected columns. Linux: the selected column's background color will be different from the background color of non-selected columns.)
  To stop filtering the table data and view the data for all I/Os, simply clear/empty the **Filter:** textfield.

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: The I/O instances in the table may be dragged from the table to sites in the Floorplanner view (see page 178) or balls in the Package View (see page 226). This will assign the dragged instance to the drop location's site/ball. The I/O's location will immediately be updated in all views.

  Changing I/O placement will require re-running Place and Route.

- Editing individual tablecell values (only available in certain columns): To edit the value displayed in a tablecell, select the cell and left-click or press Enter. The cell then becomes editable, presenting the user with a drop-down combo box of choices, or with an in-place editable textfield for free-form text entry. Change the value as desired and hit Enter to commit the change. The changed value will be validated, and if acceptable, the table will be updated to reflect the new value. Note that when the user is presented with a combobox, the choices listed will be 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: If the user wishes to change the configuration of multiple I/O instances at once (like for a bus), the user can select all the rows they wish to edit, then select the **Configure Selected IOs** button in the view toolbar. This will bring up the Configure Selected IOs Dialog (see page 266), allowing the user to edit the configuration of the entire group of I/O instances.

Table 84: *IO Assignment View Toolbar Buttons*

Icon	Action	Description
n/a	Filter:	This textfield allows the user to enter a search filter. The displayed contents of the IO Assignment table will be restricted to those rows with data matching the filter. Text matching the filter will be highlighted in the chosen column. Clear/empty the textfield 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, left-click the table's column header.)
	Configure Selected IOs	Opens the Configure Selected IOs Dialog (see page 266). Allows the user to change 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 280). Allows the user to save 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 501).
	Generate Pin Assignment Report	Opens the Generate a Pin Assignment Report Dialog (see page 274). Allows the user to generate a Pin Assignment Report (see page 322) with the same column configuration as the current IO Assignment table.

Icon	Action	Description
	Configure Columns	Opens the Configure Table Columns Dialog (see page 267). Allows the user to change which columns are shown in the IO Assignment table, allows the width of each column to be edited, and allows the user to save and load favorite configurations of this table.

Table 85: *IO Assignment Table Columns*

Column	Editable	Description
Port Name		The top-level port name in the user design used to reference this IO.
Direction		The direction of this IO: **IN**, **OUT**, or **INOUT**.
Pad/Macro		The instance name of the IO buffer (pad) or the instance name of the IO macro (a SerDes for example) in the user design.
Placed		Indicates if this IO is placed or not.
Bank		The name of the physical IO bank (or group) in which this IO is placed.
Ball	Yes	The package ball name upon which this IO is placed. 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.
Device Port		The top-level port name on the device to which this IO is connected. (Only applies if an IO is placed).
Pad/Macro Site		The site name on the device within which this IO's pad or macro instance is placed.
Site Polarity		The LVDS polarity of the site this IO is placed on. **P** = Positive and **N** = Negative.
Bank Use		The usage type of the bank this IO is placed in. **C** = Clock Bank, **G** = General Purpose Bank, **B** = Byte Lane Bank.
Clock Capable		Indicates whether the site this IO is placed on supports Clock function or not.
Reset Capable		Indicates whether the site this IO is placed on supports Reset function or not.
Data Capable		Indicates whether the site this IO is placed on supports Data signals or not.

Column	Editable	Description
I/O Standard	Yes	The IO Standard value for this IO. The IO Standard controls the voltage level and other settings for this IO. This can be changed in ACE to a different IO Standard value, but only if a compatible IO standard exists that would not require re-synthesizing the design.
Voltage Level		The VDD voltage level for this IO, set via the IO Standard selection.
VREF Level		The VREF voltage level for this IO, set via the IO Standard selection.
Pad Polarity		The LVDS polarity of this IO instance in the user design. This only applies for differential IO buffers and is set in synthesis.
Pad DQ Capability		The DQ capability of this IO instance in the user design. This only applies for byte lane IOs.
Site DQ Capability		The DQ capability of the site this IO instance is placed on. This only applies for byte lane IOs.
Site DQ Domains		The list of accessible DQ Domains that can be reached from the site this IO instance is placed on. This only applies for byte lane IOs.
Drive Strength	Yes	The drive strength of this IO. This only applies for outputs and inouts and a limited set of IO Standards.
Hysteresis	Yes	The hysteresis setting of this IO. This only applies for inputs and inouts and a limited set of IO Standards.
Location		The location property can be specified in the user's RTL to do pre-placement for this IO. 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 IO. This only applies for outputs and inouts.
Slew	Yes	The slew rate of this IO. This only applies for outputs and inouts and a limited set of IO Standards.
On Die Termination	Yes	Indicates whether on die termination is enabled for this IO. This only applies for inputs and inouts.
Termination Value	Yes	The termination value of this IO. This only applies for inputs and inouts and may have different ranges for different IO Standards. On Die Termination must be turned on for this to apply.
Keep Mode	Yes	The keep mode setting of this IO. This only applies for inputs and inouts.
Used as Clock		Indicates whether this IO is used as a clock in the design or not.

Column	Editable	Description
Used as Reset		Indicates whether this IO is used as a reset in the design or not.
Clock Domain		The clock domain for this IO.
Target Frequency (MHz)	Yes	The target frequency for the clock domain for this IO. This is set via the SDC timing constraints, or is the HW limit if unspecified.

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 will change to reflect the selected IP's configuration.

Some IP will support multiple pages of diagrams (for example, a logic block diagram page and a placement diagram page). In these cases, there will be multiple labeled tabs at the bottom of the IP Diagram view allowing the user to switch diagram pages.

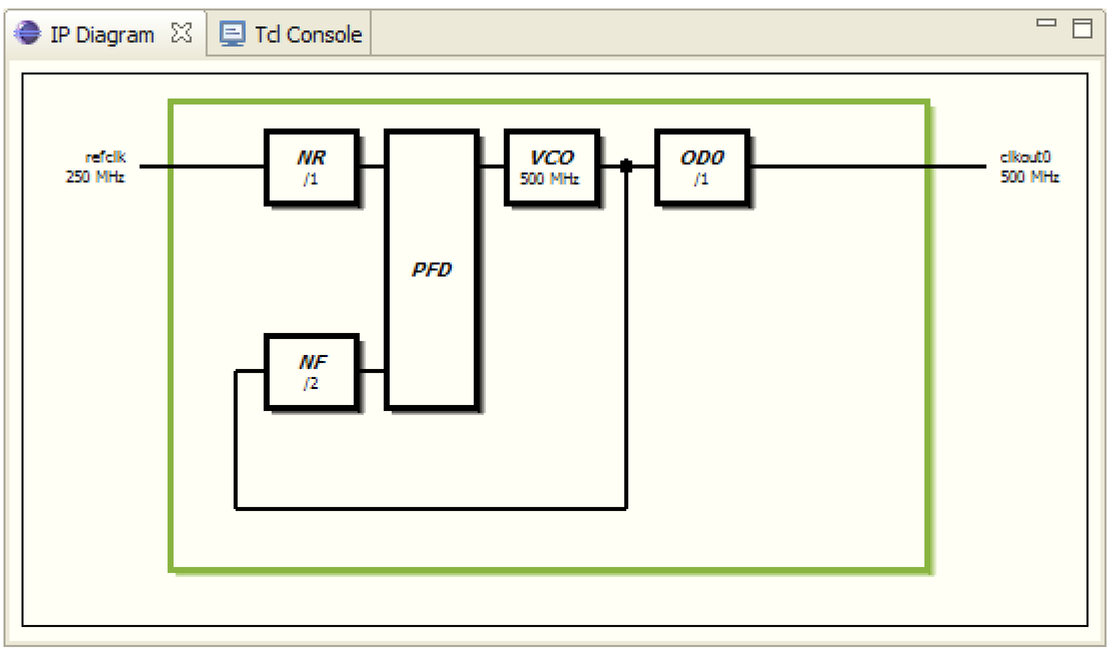
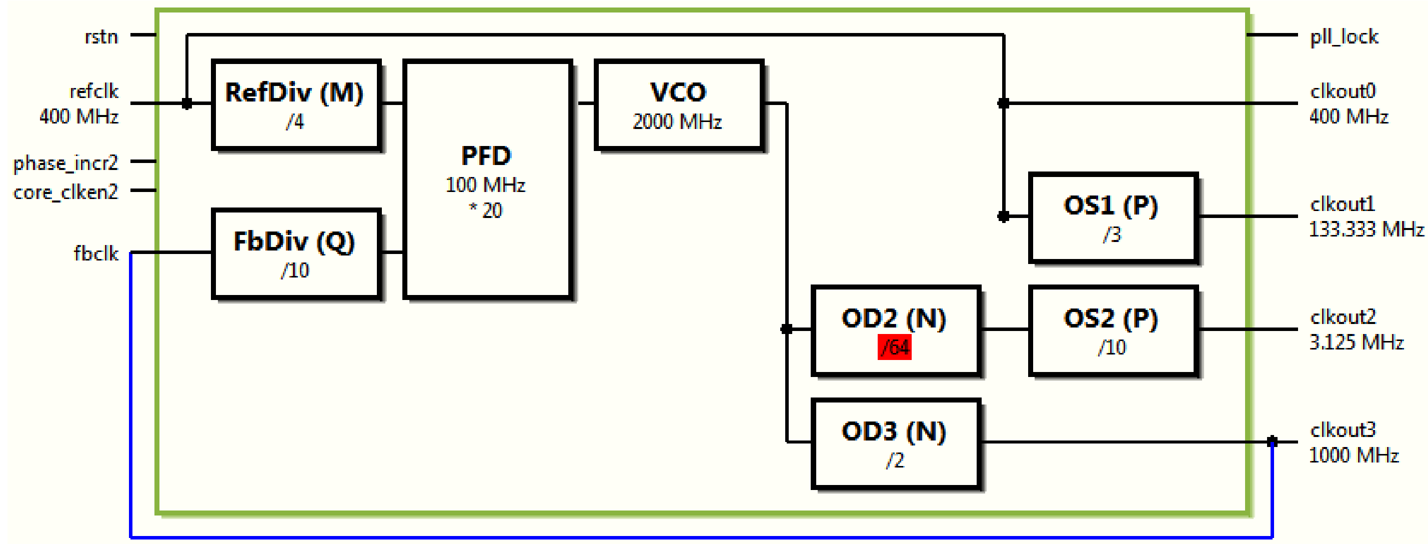


Figure 22: *Screenshot of an Example IP Diagram*

When a supported IP Configuration Editor is selected, the IP Diagram view shows a dynamic block diagram of the selected IP. Displayed labels will 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 will be 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 300).)

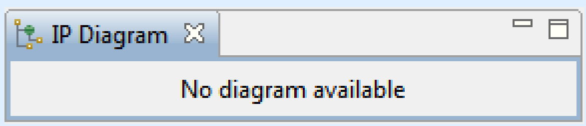


The user may left-click on any text label in the IP Diagram to immediately turn the IP Editor to the associated page so that the user may edit the related Configuration Options.

There are a number of preferences available allowing visual customization (colors and fonts) of the IP Diagram view - these are changed on the IP Diagram Preference Page (see page 300).

See also: Creating an IP Configuration (see page 387)

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 will display a notice that there is no diagram available for the selected Editor.



IP Libraries View

The IP Libraries view provides an alternate method for creating IP configuration files (`.acxip`) 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 277).

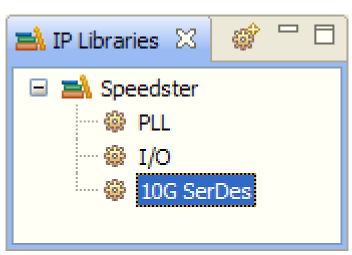


Figure 23: ***Screenshot of IP Libraries View***

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 by the ACE end user.

Table 86: *IP Libraries Toolbar Buttons*

Icon	Description
	Opens the New IP Configuration Dialog (see page 277) so the user may create a new IP configuration file.

See also: Creating an IP Configuration (see page 387)

IP Problems View

The IP Problems view displays all the warnings and errors for all 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 file (`*.acxip`), 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 on an error or warning opens the relevant IP Configuration Editor to the appropriate page. (See also Creating an IP Configuration (see page 387))

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

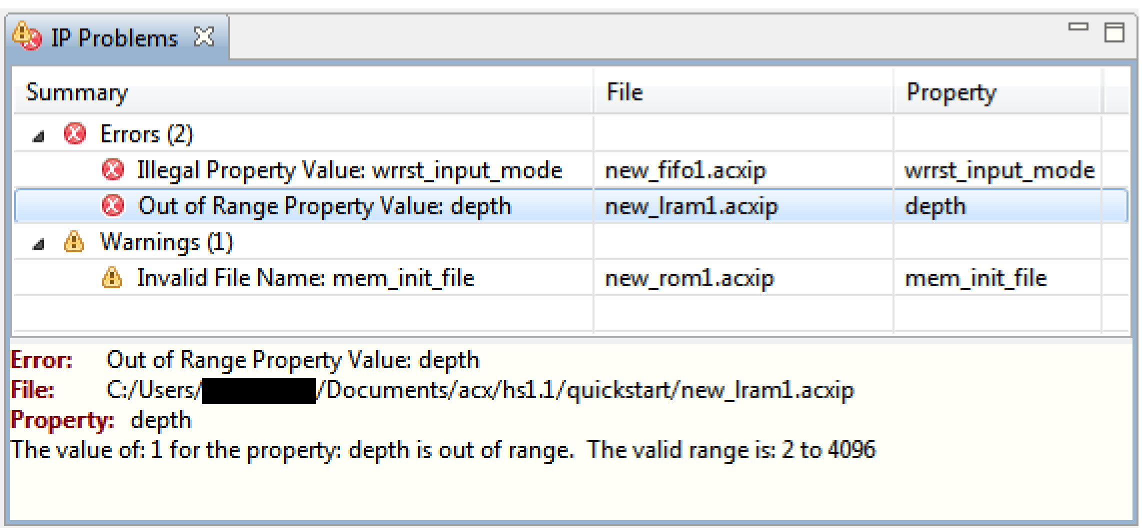


Figure 24: *Example Screenshot of the IP Problems View*

Table 87: ***IP Problems View Icons***

Icon	Description
	Warning
	Error

Table 88: ***IP Problems View Table Columns***

Column Name	Description
Summary	A brief summary statement of the IP Configuration problem.
File	The IP Configuration file (`.acxip`) 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 usually are 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 will show a much more user-friendly representation of the same data.)

JTAG Browser View

The JTAG connection must be configured before using the JTAG Browser!

ACE interacts with the FPGA using the JTAG interface through a Bitporter pod or FTDI FT2232H JTAG device. This JTAG interface must be properly configured in ACE before using the JTAG Browser functionality. The configuration is managed using the Configure JTAG Connection Preference Page (see page 288). See Configuring the JTAG Connection (see page 411) for more details.

Check clock configuration.

Some IP Blocks will require a connected, running SBUS clock before they'll work with the JTAG Browser functionality. The SerDes IP blocks must have connected reference clocks, which must be free running even if you are not debugging that particular lane.

Please talk to your FAE or Achronix Tech Support for further important details, including the required SBUS /reference clock frequency ranges for your target device / IP block combination.

The JTAG Browser view provides the user with an interactive means of inspecting and modifying registers within the active design on an FPGA over the JTAG interface. (The acx_stapl_player and Bitporter pod or FTDI FT2232H JTAG device perform the JTAG interactions; see the *Bitstream Programming and Debug User Guide (UG004)* for more information.)

After choosing the Target Device and the IP Block within that device, the user is able to browse and edit registers on a live device. All accessible IP blocks on the FPGA are selectable from a pull down list; once selected, the attributes (base-address, end-address, word-size, etc) for the selected IP block are displayed, along with the acx_stapl_player commands which will be used to read and write to the block's registers.

By default, the JTAG Browser view is included in the Programming and Debug Perspective (see page). To add it to the current perspective, click **Window -> Show View... -> JTAG Browser**.

When the JTAG Browser view opens, the windows may need to be resized for optimal viewing.

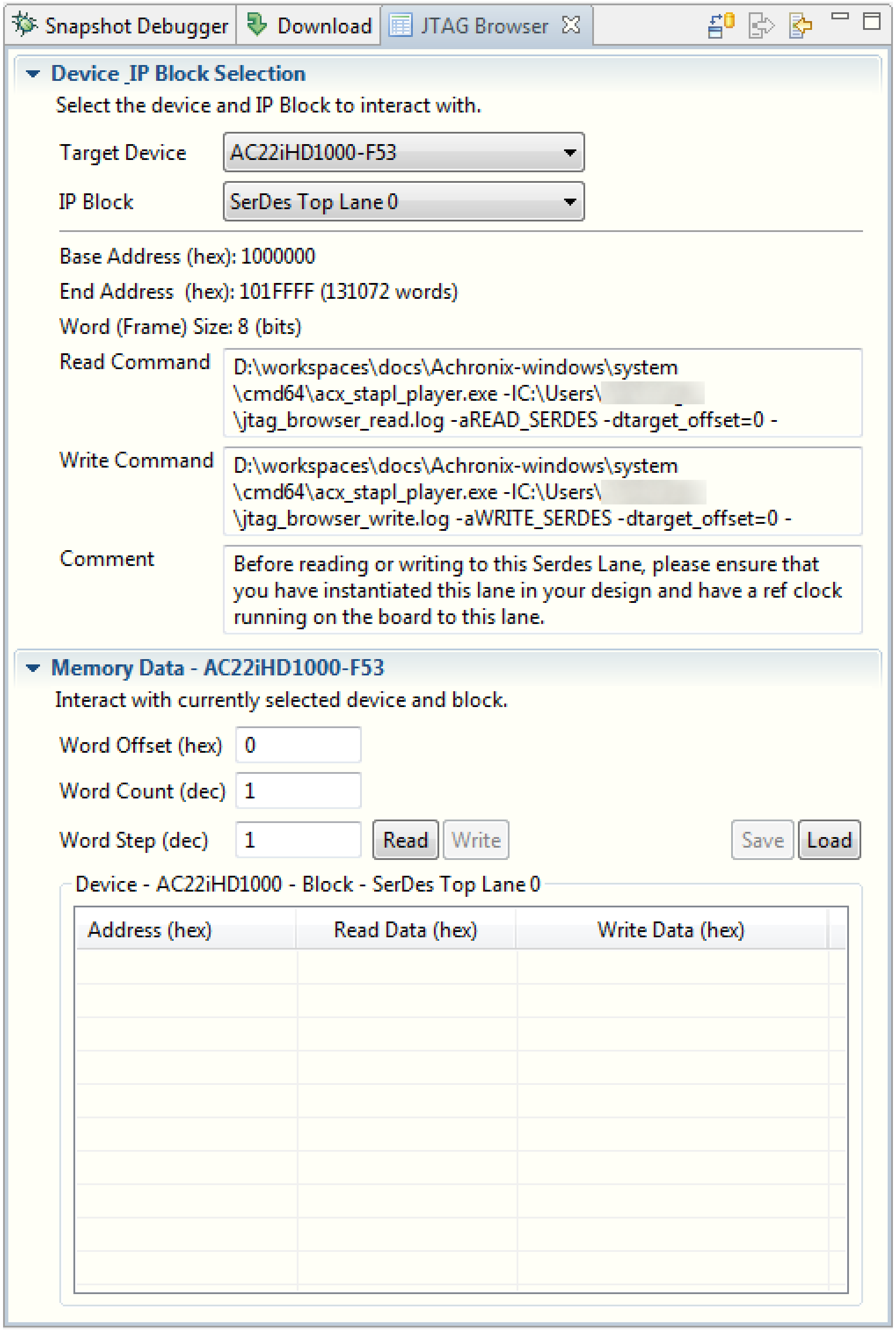

Snapshot Debugger
Download
JTAG Browser
Device_IP Block Selection
Select the device and IP Block to interact with.
Target Device
AC22iHD1000-F53
IP Block
SerDes Top Lane 0
Base Address (hex): 1000000
End Address (hex): 101FFFF (131072 words)
Word (Frame) Size: 8 (bits)
Read Command
D:\workspaces\docs\Achronix-windows\system
\cmd64\acx_stapl_player.exe -lC:\Users\
\jtag_browser_read.log -aREAD_SERDES -dtarget_offset=0 -
Write Command
D:\workspaces\docs\Achronix-windows\system
\cmd64\acx_stapl_player.exe -lC:\Users\
\jtag_browser_write.log -aWRITE_SERDES -dtarget_offset=0 -
Comment
Before reading or writing to this Serdes Lane, please ensure that
you have instantiated this lane in your design and have a ref clock
running on the board to this lane.
Memory Data - AC22iHD1000-F53
Interact with currently selected device and block.
Word Offset (hex)
0
Word Count (dec)
1
Word Step (dec)
1
Read
Write
Save
Load
Device - AC22iHD1000 - Block - SerDes Top Lane 0
Address (hex)
Read Data (hex)
Write Data (hex)

Table 89: *JTAG Browser View Actions*

Icon	Action	Description
	Configure JTAG Interface	Opens the ACE preferences dialog with the Configure JTAG Connection Preference Page (see page 288) selected and visible.
	Save	Displays the Save File dialog, allowing the user to save the current view contents (specifically the Target Device and IP Block fields, and Memory Data table's Address and Read Data columns) to a specified text file for later reference.
	Load	Displays the Load File dialog, allowing the user to select and load previously saved (read) configuration data from a text file into the view (specifically the Target Device and IP Block fields, and Memory Data table's Address and Write Data columns).
	Read	When pressed, reads† the specified data range from the active design using JTAG. Results of the read operation are used to populate the table. The resulting log file is opened for viewing.
	Write	When pressed, writes† any changed addresses to the active design using JTAG. The Write action will not be enabled unless / until read data is present and at least one row has been modified in the table.
† Any errors reported by the acx_stapl_player during the read or write operations will be visible in the log file produced by the Read or Write operation. This log file is automatically opened for viewing when the read or write command completes.		

Table 90: *JTAG Browser View Fields*

Field	Editable	Description
Device IP Block Selection		
Target Device	✓	Drop down list of device + package options.
IP Block	✓	Drop down list of IP blocks available with the current device / package selection.
Base Address		Beginning address of registers for the selected IP block.
End Address		Ending address of registers for the selected IP block.
Word Size		Width in bits of selected block's registers.
Read Command		The `acx_stapl_player` command which will be used to read data from the selected IP block.

Field	Editable	Description
Write Command		The `acx_stapl_player` command which will be used to write data to the selected IP block.
Memory Data		
Word Offset	✔	Number of words from 'base address' to start reading from.
Word Count	✔	Number of words to read.
Word Step	✔	Address step size between words when doing a read operation
Address		Register address associated with table row.
Read Data		Original data value read from address for table row.
Write Data	✔	User modified data value for row address; initially the same as Read Data value. Changed values will be highlighted.

Saving Register Data To a File

Once a range of addresses has been read into the view's data-table, the values in the table can be saved to a text file in the following formats:

- CSV (comma separated values)
- HTML
- XML

This text file is convenient for saving interesting results, for sharing with colleagues, or for later reference. It can also be re-loaded into the active design for initialization purposes.

Loading Register Data From a File

Loading (previously-saved) register data is a little more involved than saving it. Once you have selected a file to load, the range of register addresses is calculated and a read-operation is performed to refresh the data-table. If no errors occur during the refresh, the saved write-data values from the file are applied to the data-table; any rows with differing read and write data values are highlighted. At this point the **Write** button is enabled and you may click on it to modify write the changed register values to the device.

Note that the IP block associated with the saved data-file will be made active ('auto-selected') when the file is loaded.

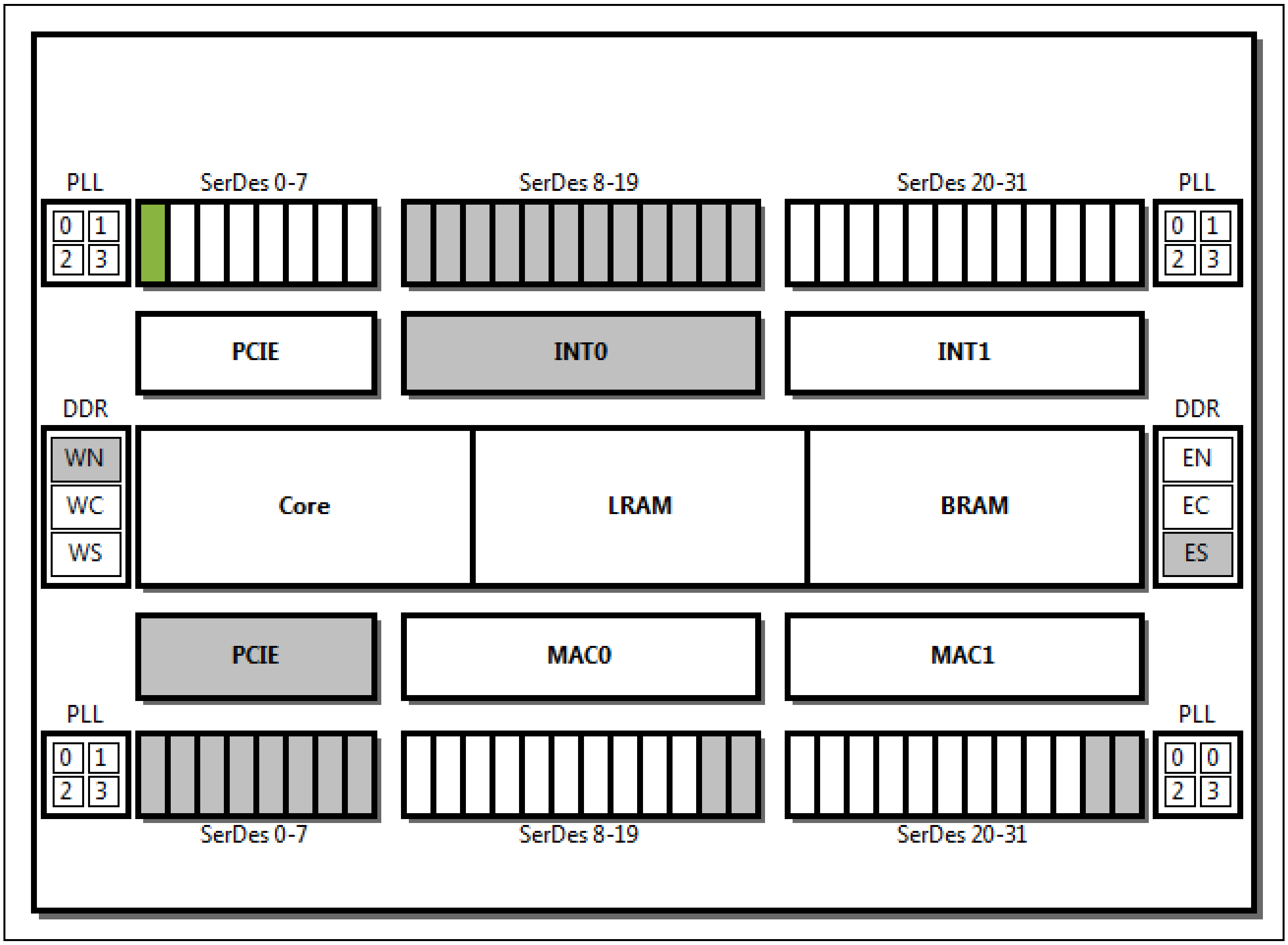
JTAG Diagram View

The JTAG Diagram view works with the JTAG Browser View (see page 199), and provides the user with a convenient and interactive means of selecting IP blocks for further inspection. Clicking on a block will cause it to be selected in the JTAG Browser View (see page 199) with all of its attributes displayed.

Additionally, the diagram shows the user which blocks are available in the currently selected device/package combination. Gray blocks are unavailable, clear (or white) blocks are accessible; the green block is currently selected for inspection.

By default, the JTAG Diagram view is included in the Programming and Debug Perspective (see page). To add it to the current perspective, click **Window -> Show View... -> JTAG Diagram**.

When the JTAG Diagram view opens, the windows may need to be resized for optimal viewing.



Multiprocess View

Similar to the Tcl command run_multiprocess (see page 583), the Multiprocess View ( ) allows Running Multiple Flows in Parallel (see page 363) and Attempting Likely Optimizations Using Option Sets (see page 434).

The Multiprocess View provides a way for users to select multiple Implementations (see page 312) within a single project (see page 312) 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 the user explore the solution space provided by various ACE optimizations. The Multiprocess View is able to optionally generate new implementations derived from the current Active Project and Implementation (see page 316), where each newly generated implementation applies an overlay of likely implementation option (see page) optimizations over the active implementation's options. These collections of potentially optimized implementation options are termed option sets (see page).

By default, the Multiprocess View is hidden, but is considered a part of the Projects perspective (see page 23). The Flow View (see page 187) (which is also part of the Projects perspective) has a button ( ) in its button bar which will cause the Multiprocess View to be shown – when pressed, this button will minimize the ACE Editor Area (where reports are displayed) and will show the Multiprocess View.

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", and "Multiprocess Run Logs". Each section includes a brief descriptive paragraph describing its purpose. Each section may be collapsed and expanded by clicking on the section title. Collapsing or expanding any section will cause the other sections to be re-sized to fit the available data and view area.

For more detailed info on how to use this view, please see Running Multiple Flows in Parallel (see page 363) and Attempting Likely Optimizations Using Option Sets (see page 434).

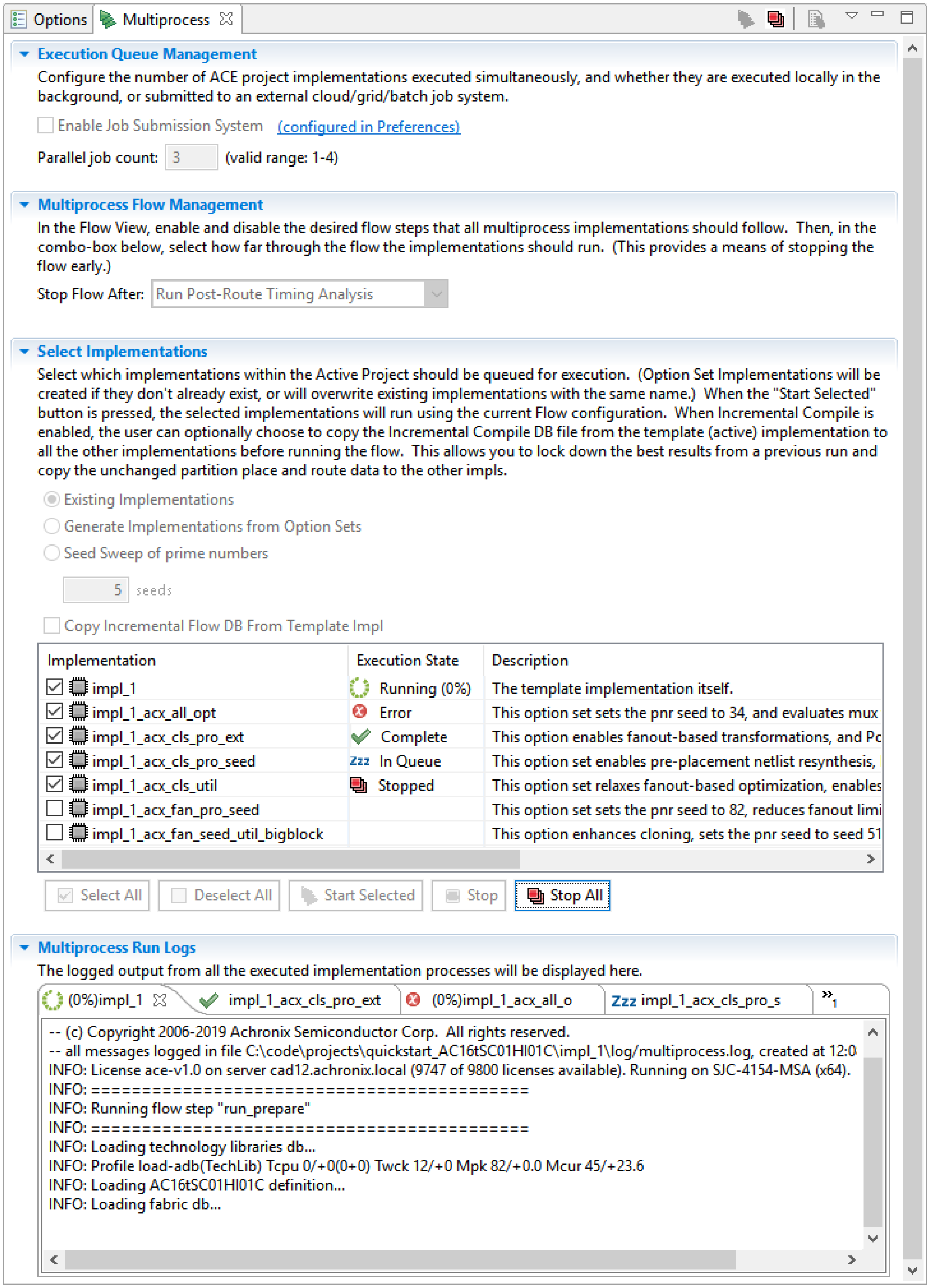

Options
Multiprocess
Execution Queue Management
Configure the number of ACE project implementations executed simultaneously, and whether they are executed locally in the background, or submitted to an external cloud/grid/batch job system.
Enable Job Submission System
(configured in Preferences)
Parallel job count: 3 (valid range: 1-4)
Multiprocess Flow Management
In the Flow View, enable and disable the desired flow steps that all multiprocess implementations should follow. Then, in the combo-box below, select how far through the flow the implementations should run. (This provides a means of stopping the flow early.)
Stop Flow After: Run Post-Route Timing Analysis
Select Implementations
Select which implementations within the Active Project should be queued for execution. (Option Set Implementations will be created if they don't already exist, or will overwrite existing implementations with the same name.) When the "Start Selected" button is pressed, the selected implementations will run using the current Flow configuration. When Incremental Compile is enabled, the user can optionally choose to copy the Incremental Compile DB file from the template (active) implementation to all the other implementations before running the flow. This allows you to lock down the best results from a previous run and copy the unchanged partition place and route data to the other impls.
Existing Implementations
Generate Implementations from Option Sets
Seed Sweep of prime numbers
5 seeds
Copy Incremental Flow DB From Template Impl
Implementation
Execution State
Description
impl_1
Running (0%)
The template implementation itself.
impl_1_acx_all_opt
Error
This option set sets the pnr seed to 34, and evaluates mux
impl_1_acx_cls_pro_ext
Complete
This option enables fanout-based transformations, and Pc
impl_1_acx_cls_pro_seed
In Queue
This option set enables pre-placement netlist resynthesis,
impl_1_acx_cls_util
Stopped
This option set relaxes fanout-based optimization, enables
impl_1_acx_fan_pro_seed
This option set sets the pnr seed to 82, reduces fanout limi
impl_1_acx_fan_seed_util_bigblock
This option enhances cloning, sets the pnr seed to seed 51
Select All
Deselect All
Start Selected
Stop
Stop All
Multiprocess Run Logs
The logged output from all the executed implementation processes will be displayed here.
(0%)impl_1
impl_1_acx_cls_pro_ext
(0%)impl_1_acx_all_o
impl_1_acx_cls_pro_s
-- (c) Copyright 2006-2019 Achronix Semiconductor Corp. All rights reserved.
-- all messages logged in file C:\code\projects\quickstart_AC16tSC01HI01C\impl_1\log/multiprocess.log, created at 12:0
INFO: License ace-v1.0 on server cad12.achronix.local (9747 of 9800 licenses available). Running on SJC-4154-MSA (x64).
INFO: ==
INFO: Running flow step "run_prepare"
INFO: ==
INFO: Loading technology libraries db...
INFO: Profile load-adb(TechLib) Tcpu 0/+0(0+0) Twck 12/+0 Mpk 82/+0.0 Mcur 45/+23.6
INFO: Loading AC16tSC01HI01C definition...
INFO: Loading fabric db...

Table 91: ***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 323) 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 92: ***Execution Queue Management Controls***

Name	Description
Parallel Job Count	Sets the number of implementations which will be allowed to execute in parallel. Defaults to **2**. When in background mode, the maximum allowed value will be the number of available processors detected. When in Job Submission System mode, the maximum allowed value will be 99.
Enable Job Submission System Support	When unchecked, background processes will run locally on the workstation currently running the ACE GUI. When checked, ACE will use the cloud/grid/batch job submission system as configured in the preferences.
(configured in Preferences)	This link, when selected, will bring up the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 302), allowing the user to fully configure which cloud/grid/batch job submission system will be used.

When the **Parallel Job Count** is set to the minimum value of **1**, all selected implementations will be executed sequentially, one at a time. A value of **2** would cause all selected implementations to be queued, and then the first two queued implementations would be 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 323) 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. This is termed "background mode". In this case, the effectiveness of parallel implementation execution will naturally be limited by the resources of the host workstation.

Alternately, ACE may execute the implementations in processes distributed among multiple hosts via an external job submission system, which will theoretically allow 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 302), reached easily by following the ***(configured in Preferences)*** hyperlink.

See Running Multiple Flows in Parallel (see page 363) for important details regarding parallel implementation execution, configuration, and external job submission tool support.

Multiprocess Flow Management

This section allows the user to alter how far the flow will be executed for the multiprocess implementations.

Table 93: *Multiprocess Flow Management Controls*

Name	Description
Stop Flow After	Allows the user to override standard flow behavior, and stop 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 the user still wishes to run multiple implementations through earlier parts of the flow. **Note** The flow step chosen here will be always be 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 363) for further details regarding multiprocess flow configuration.

Select Implementations

This section allows the user to select which implementations they wish to execute (implementations derived from option sets (see page) will be created if selected), start or stop the execution of all selected implementations, and provides simple execution state feedback.

See Running Multiple Flows in Parallel (see page 363) 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 434) for explanations of how to use option sets to achieve better QOR.

Table 94: *Select Implementations Controls*

Name	Description
Existing Implementations	This radio button will update the contents of the Implementation Table to show all existing implementations for the current active project (see page 316).
Generate Implementations from Option Sets	This radio button will update the contents of the Implementation Table to show the current active implementation (see page 316) and a number of to-be-generated implementations, one per Option Set (see page). The use of this radio button selection is covered in more detail at Attempting Likely Optimizations Using Option Sets (see page 434).
Seed Sweep of prime numbers	This radio button will update the contents of the Implementation Table to show a number of to-be-generated implementations. Each of these implementations will be identical to the currently active implementation, with the implementation option 'seed' being automatically set to the next consecutive prime number. The text field below 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 will vary based upon the active project and implementation (see page 316), in combination with the state of the radio buttons.
Select All	This button causes all implementations listed in the Implementation Table to become selected.

Name	Description
Deselect All	This button causes all implementations listed in the Implementation Table to become deselected.
Start Selected	Causes all implementations selected in the Implementation Table to queue up and begin executing in the configured number of parallel processes.
Stop All	If pressed, all currently queued implementations are removed from the queue(s) and all currently executing implementations are killed. The Multiprocess Summary Report (see page 323) will be updated with any and all captured information.

All the controls in this section center around what's 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 316). Changing active projects (which implicitly changes active implementations) will update the Implementation Table contents according to the current radio button selection.

The Implementation Table's columns are each described below.

Table 95: *Implementation Table Columns*

Column Name	Description
Implementation	This column contains the implementation's name, along with a checkbox indicating implementation selection, and an icon representing the implementation.
Execution State	This column contains the execution state of the implementation.
Description	Blank when **Existing Implementations** is selected. When **Generate Implementations from Option Sets** is selected, this column contains a description of the Option Set (see page) which will be used as the overlay on the active implementation (see page 316) when generating the new implementation.

Tip

If the implementation table isn't 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 will cause 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's background process. The icons from these states are also used on the tabs within the Multiprocess Run Logs section.

Table 96: *Implementation Execution States and Icons*

Icon	Execution State	Description
no icon	*blank*	This implementation has not been selected for execution.
no icon	Selected	This implementation is currently selected for execution, and execution has not been started.
	In Queue	Execution of the selected implementations has been started, this implementation was selected for execution, and this implementation is currently waiting in the queue for execution.
	Scheduled	Execution of the selected implementations has been started, this implementation was selected for execution, and this implementation 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 current executing. Log messages for this implementation 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 for this implementation should be visible in the tabbed logging area. Summary information for this implementation should be visible in the Multiprocess Summary Report (see page 323).
	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 for this implementation 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 323).
	Error	This implementation was (and still is) selected for execution, but its last execution exited with reported errors. A tooltip for the error icon will provide a summary of the detected error messages. Detailed log messages for this implementation 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 323).

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 will include 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 will be updated to include a summary of the captured error messages. Error details will be visible in the log shown in the tab, as well as within the Log Files (see page 315) for each implementation.

Netlist Browser View

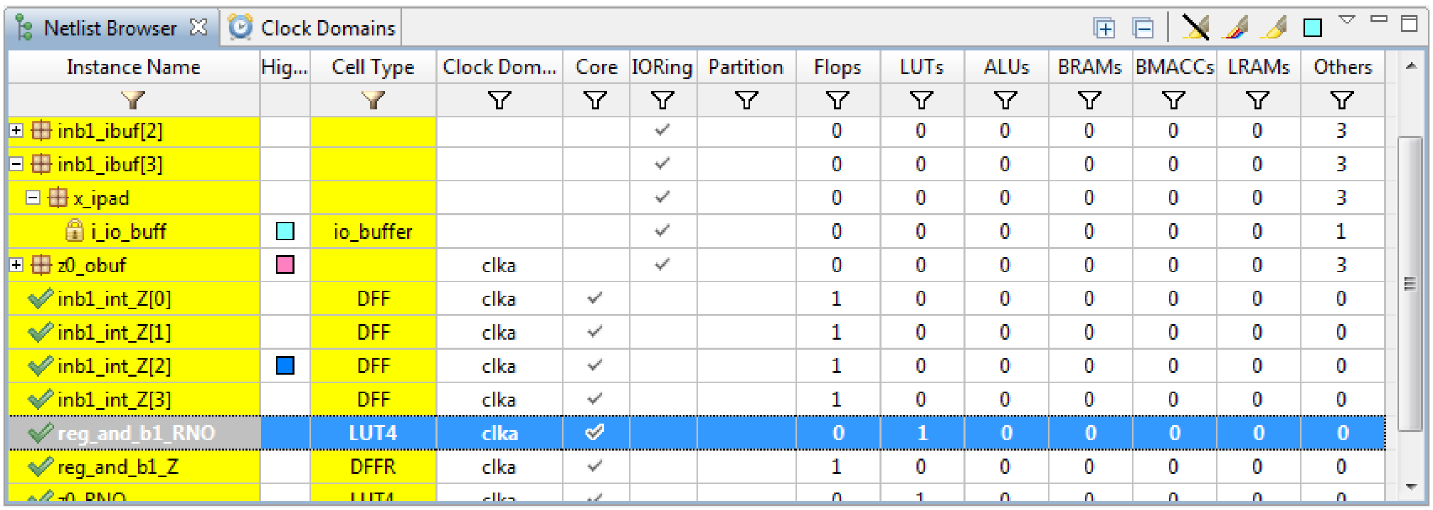
The Netlist Browser view's purpose is to provide a graphical, tree-based visualization of the user's design hierarchy, as found in the netlist. The displayed netlist includes the results of any transformation, legalization, etc. that have happened through the current stage in the Flow.

For large designs, there are a tremendous number of objects in the netlist. To reduce the visual noise, 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 will include the instance's name and the cell type. Macros will include the macro's 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 will always represent the unfiltered total counts. Clock domain names and Partitions names will also be listed when appropriate.

By default, the Netlist Browser view is included in the Floorplanner perspective (see page 23). To add it to other perspectives, select **Window → Show View... → Other... → Achronix → Netlist Browser**.

As can be seen below in the second column, 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, and `inb1_int_z[2]` in dark blue. Also note that the small colored square in the toolbar shows the active highlighting color. If highlighting is applied to a macro then all “child” instances within will also be set to the current highlight color.



> **Note**
>
> 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 by the ACE end user.

Table 97: *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.

Column Name	Description
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 will show a color square for that same highlight color. If even one contained instance has a different highlight color, or no highlight at all, then the macro will display no color square. This value will not change for macros during filtering.
Cell Type	For Instances, shows the cell type of the instance. For Macros, this column will be 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 will not change for macros during filtering.
Core	For Instances, this will be checked if the instance is considered a member of the Core, or blank if it is not. For Macros, this will be checked if any contained instances are considered a member of the Core, or blank if no contained instances are in the Core. This value will not change for macros during filtering.
IORing	For Instances, this will be checked if the instance is considered a member of the IORing, or blank if it is not. For Macros, this will be checked if any contained instances are considered a member of the IORing, or blank if no contained instances are in the IORing. This value will not change for macros during filtering.
Partition	For Instances and Macros, the name of the Partition to which the item belongs, if any. See Using Incremental Compilation (Partitions) (see page 443)
Resource	For Instances, this will be one if the instance is of type *resource*, or zero otherwise. For Macros, this will be the sum count of all contained *resource* instances (regardless of filtering).

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

Table 98: *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 and (right-click) context menus on the nodes of the tree. Note that if these actions are performed upon macros, all child leaf nodes, even those currently filtered to be hidden in the tree, will be affected by the chosen action.

Table 99: ***Netlist Browser View Actions***

Icon	Action	Toolbar Button	Context Menu	Description
	Add to Selection		Y	Adds the item(s) to the ACE Selection Set (as shown in the Selection View (see page 249)).
	Remove from Selection		Y	Removes the item(s) from the ACE Selection Set (as shown in the Selection View (see page 249)).
	Choose Highlight Color	Y	Y	Determines which color will be 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 396).
	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) will 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 will be skipped.
	Zoom To		Y	Zooms the Floorplanner view and Package view to a region containing the instances currently chosen in the tree.
	Show in Netlist		Y	Attempts to open a text editor to the file and line number relevant to the chosen instance. Available only when a single instances is chosen in the view. *Note: this is Early Access functionality; this may not always open the text editor to the expected location.*
	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.

Icon	Action	Toolbar Button	Context Menu	Description
	Toggle Filter Row Visibility		Y	Changes whether the filter row (of filter icons) is visible or not. Note that this does not alter whether filters are active, it only changes the visibility of the row of filter icons.
	Configure view...		Y	Jumps to the Netlist Browser view in the Preferences dialog

Actions upon macros affect all children

Be aware that when actions are performed upon macros, all the children of that macro, even the invisible /filtered nodes, are affected.

Reminder: Instances' Selection color vs Highlight color priority

With default preference settings, in the Floorplanner View (see page 178), Highlight colors of (placed) instances will only be visible when the Instances Layer is enabled, and the instances are not members of the ACE Selection Set. This is because the Instance's Selection color has a higher priority than the Highlight color.

Filtering Displayed Instances

To enable instance filter manipulations, users may need to **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's 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 may filter the displayed instances (not the macros) by value. When filtering by column value, only instances with column values matching the filter will be retained; non-matching values will be excluded from the table.

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

Column filters are applied to instance rows, not macros.

When using filters, the values being filtered are those of the individual instances, not the macros. Macros will be filtered out only if all of their children are filtered out.

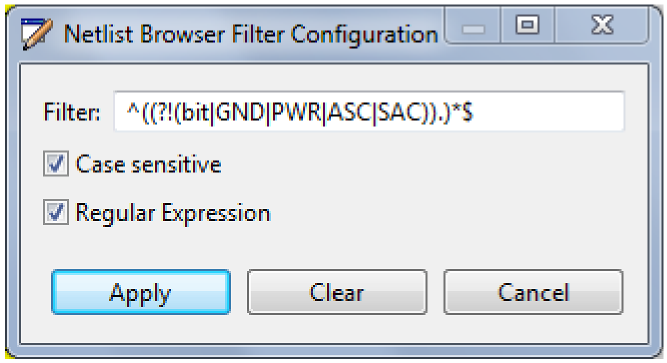
This means that when filtering by the logic types, the only possible filter numeric values in this table will be 0 or 1, because these are the only legal values for an instance.

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 (glob string matching by default, but Regular Expression matching is also available, following Java rules, 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.

To add a filter to a column, simply left-click the mouse on the filter icon ( ), which causes a data-appropriate filter dialog to appear. Then fill in the desired filter values and press **Apply** to apply the filter to the instances in the table. All values matching that filter will be retained, and all other values will be excluded. Additionally, the background color of the column will change to a bright yellow to indicate the filter is active, and the filter icon at the head of the column will also change 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.



To edit (or clear) an existing filter, simply left-click the mouse on the active filter icon ( ), which again causes the data-appropriate filter dialog to appear, this time pre-populated with the existing filter setting. Change the filter value and press **Apply** again to edit the filter, press **Cancel** to leave the filter unchanged, or press **Clear** to remove the filter from the column. If the filter is cleared, the background color of the column will return to the default background color, and the filter icon will also change 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 23). The Netlist Browser view only acts as a Drag-and-Drop source; items dropped on the Netlist Browser view will be ignored.

Any node of the tree may be dragged to the Tcl Console view (see page 258), 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 178) or Package view (see page 226). When dropped on the Package view, placement of the instance (see page 399) is attempted at the specified site. When dropped on the Floorplanner view, the behavior depends upon the current Tool mode. When the Floorplanner's **Placement/Panning Tool** is active, placement is attempted.

Any node of the tree may be dragged to the Placement Regions view (see page 234) or the Floorplanner view (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 438). Dragging a macro is the equivalent of dragging all individual instances which are members of that macro. Be aware that since placement regions may only encompass the fabric core, any dragged I/O instances are not assigned to placement regions.

Options View

The Options view displays project (see page 312) implementation options (see page) for the active implementation (see page 312). From this view, the active project implementation (see page 316) can be configured for its run through the flow (see page 316).

This view does not display any information unless an active implementation is selected in the Projects View (see page 238). When Running the Flow (see page 361), 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 23). To add it to the current perspective, select **Window** → **Show View...** → **Options**.

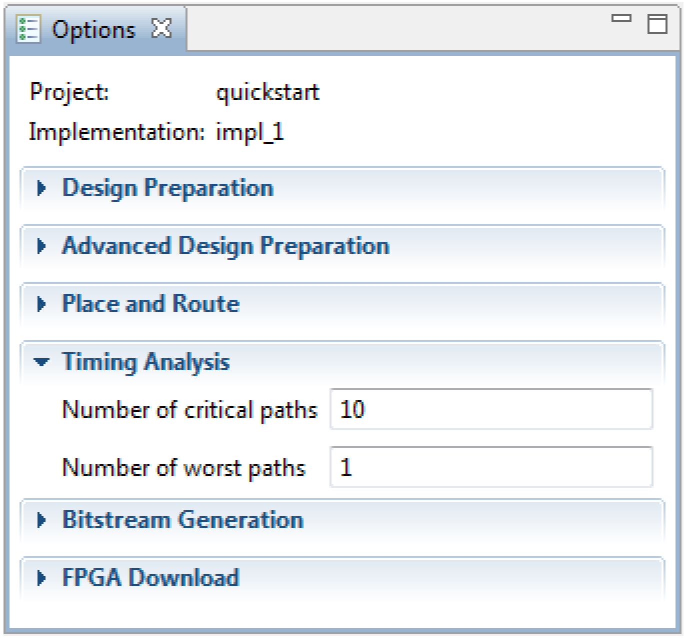


Figure 25: ***Screenshot of the Options View***

Tip: Tcl Equivalence

Each implementation option that can be configured via this graphical view may also be configured via the set_impl_option (see page 594) Tcl command. The current value of each option can be retrieved with the get_impl_option (see page 558) Tcl command.

The values of options may be reset back to their default values with the reset_impl_option (see page 579) 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 standard supported options used by most users, but are only a subset of all the options available to users.

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 will not be 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 325), which can be generated with the Tcl command report_impl_options (see page 575).

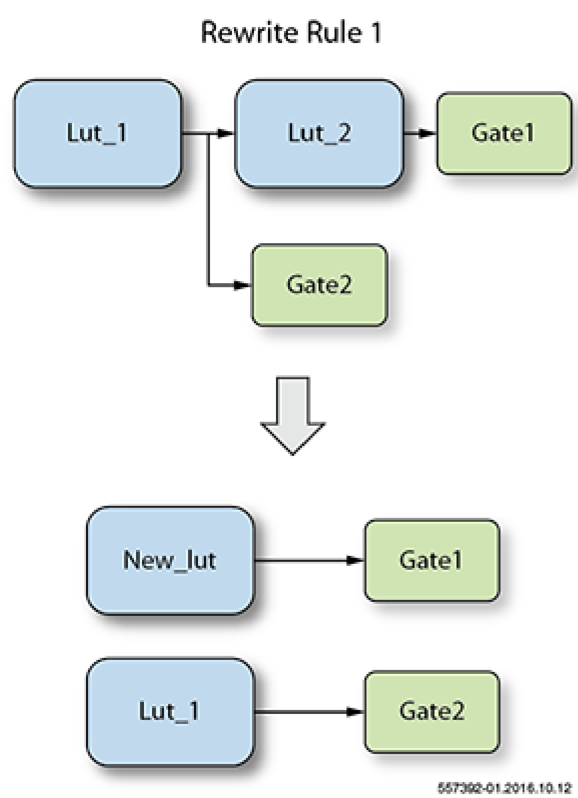
Table 100: ***Design Preparation Implementation Options***

Option	TCL Option	Description
Target Device	`partname`	This option is used to specify the name of the FPGA part to use for this implementation. **Note: 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 / common between devices. Changing the value for target device may have a ripple effect upon other option values. Thus, users may wish to review the values of all other options after changing the target device value.
Package	`package`	This option is used to specify the FPGA package for the target device.
Speed Grade	`speed_grade`	This option allows the user to select the desired speed grade for the target device.
Core Voltage	`core_voltage`	This option allows the user to select the core voltage for the target device.
Junction Temperature	`junction_temperature`	This option allows the user to select the junction temperature for the target device.
Flow Mode	`flow_mode`	**Evaluation** flow 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. **Normal** flow mode enforces all DRC checks necessary to generate a correct bitstream. Some checks are flagged as warnings early on in the flow to give the user 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** flow mode is similar to **Normal** flow mode, but enforces all DRC checks and errors out as early in the flow as possible. Bitstream generation requires **Normal** or **Strict** flow mode. See also: Flow Mode (see page 321)

Option	TCL Option	Description
Enable Incremental Compile	`incremental_compile`	Enables ACE's Incremental Compilation functionality. The upstream synthesis tool must also have have incremental compilation enabled (Synplify Pro's compile points), and the necessary constraint files must be included in the ACE project, otherwise ACE will be unable to make use of this feature as intended. See Using Incremental Compilation (Partitions) (see page 443).
Incremental Compile Mode	`incremental_compile_mode`	Incremental Compile can either be in **Strict** or **Smart** mode. **Strict** mode ensures that Placement of locked instances in unchanged partitions is completely preserved. **Smart** mode allows ACE to try to intelligently preserve placement in locked partitions for better design performance.
Export All Partitions	`export_all_partitions`	When enabled, all leaf-level partitions will be exported.
Auto-Select Top Module	`autoselect_top_module`	This option is used to specify the whether or not the top module name for this implementation should be automatically selected. A value of **1** causes the name to be automatically selected. A value of **0** causes the `-top_module` implementation option value to be used.
Top Module Name	`top_module`	This option is used to specify the top module name for this implementation when the `-autoselect_top_module` implementation option is set to **0**.
Constraint Files	Uses the `enable_project_constraints` and `disable_project_constraints` commands	This allows the user to enable or disable the use of an existing project SDC/PDC constraint file for use in this implementation's flow. All constraint files defined for the active project will be listed. Constraint files will be loaded in the order listed. To change the order constraint files are loaded, see Adding Source Files (see page 355).

Table 101: ***Advanced Design Preparation Implementation Options***

Option	TCL Option	Description
Timing-Driven Clustering	`timing_driven_clustering`	Specifies whether timing-driven clustering will be enabled during placement.
Fanout Control	`fanout_control`	When fanout control is enabled, nets with a fanout higher than the **Fanout Limit** are refactored.

Option	TCL Option	Description
Fanout Limit	`fanout_limit`	The fanout limit specifies the maximum fanout any net can have when **Fanout Control** is enabled.
Fanout Limit for Critical Nets	`critical_fanout_limit`	The fanout limit specifies the maximum fanout any net 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 will depend upon the strategies chosen below. If the “Place and Route” Implementation Option **Timing-Driven PnR** is disabled, resynthesis timing optimizations will also be disabled.
Rewrite Rule-1	`resynthesis_rewrite_rule1`	When enabled, and **Resynthesis Mode** is set to **Optimize for Timing**, this 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: 

Option	TCL Option	Description
Move Flip-flop Reset	`resynthesis_move_ff_reset`	Specifies whether resynthesis will move flip-flop reset logic to LUTs when **Resynthesis Mode** is **Optimize for Timing**.
Period of Anti-Aging Oscillator (in ns)	`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. Set this option to a value of **0** to disable insertion of anti-aging area fill logic. **Note** Setting this option to a **non-zero** value will increase the size of the user design in place and route and the corresponding bitstream to near maximum size.
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 490) contains more details.
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 can't 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 will fit in the target device.
		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 482) contains further details.

Option	TCL Option	Description
Push Flops Into Pads	`push_flops_into_pads`	**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 "`ace_useioff=0`" **Manual** – push flip-flops only into the I/O pins which have the attribute " `ace_useioff=1`"
Pad Flop Pushing Clock Type	`pad_flop_pushing_clock_type`	Control over flop pushing into IO pins by clock type. Automatic Flop Pushing into I/O Pads (see page 482) contains further details. **Boundary** – Only enable flop pushing into IO.pins clocked by a boundary clock. **Trunk** – Only enable flop pushing into IO pins clocked by a trunk clock. **All** – Enable flop pushing into all IO pins.

Table 102: ***Place and Route Implementation Options***

Option	TCL Option	Description
PnR Mode	`timing_driven_pnr`	If **Timing Driven** mode is selected, data from timing analysis will be used to optimize the design for high speed. If **Fast** mode is selected, placement and routing will be optimized for PnR runtime.
Power-Driven PnR	`power_driven_pnr`	If turned on, placement and routing will be power-driven. Data from power analysis will be used to optimize the design for low power.
Multi-Threaded PnR	`mt_pnr`	Enable Multi-Threaded Place and Route. Enabling this may speed up your compile time.
PnR Seed	`seed`	The place and route seed is used to initialize random number state in the place and route algorithms.
Placement Effort	`placement_effort`	**Low** effort placement will have 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 will attempt 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 103: ***Report Generation Implementation Options***

Option	TCL Option	Description
Report all temperature corners	`report_sweep_temperature_corners`	When enabled, ACE will loop over the valid junction temperatures for the target device and generate a separate report for each corner. See also: Timing Across All Temperature Corners (see page 331)
Output Utilization Reports	`report_utilization`	Enable utilization analysis and report generation in the flow.

Option	TCL Option	Description
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.

Table 104: ***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 will include unconstrained timing paths in the timing analysis reports.

Table 105: ***Bitstream Generation Implementation Options***

Option	TCL Option	Description
Serial Flash (`.flash`)	`bitstream_output_flash`	This option 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.

Option	TCL Option	Description
4x Serial Flash (.flash4x_0-3)	`bitstream_output_4xflash`	This option enables the generation of four additional 4x serial flash formatted output files. These files will be named the same as the STAPL file, but with a `.flash4x_0` to `.flash4x_3` extension. 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`	This option 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. It 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. It contains the same data in the same bit order as the `.cpu` file.
CPU Bus Width	`bitstream_output_cpu_width`	This option 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`	This option enables the generation of an additional Raw Hex formatted output file. This file will be named the same as the STAPL file, but with a `.hex` extension. This file is used for debug purposes.
Serial Flash Clock Divider	`bitstream_flash_clock_div`	This option selects the serial flash clock divider value
Encrypt Bitstream	`bitstream_encrypted`	Specifies whether or not the generated bitstream data will be encrypted.
256-bit Encryption Key (hex)	`bitstream_encryption_key`	Specifies the 256-bit AES encryption key used to encrypt the bitstream data. This must be specified as a 64-character hexadecimal value.
		This option specifies whether the bitstream STAPL file will be 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), then

Option	TCL Option	Description
Single Device Chain	`bitstream_single_device`	either the a chain description file will be used or the pre-IR, post-IR, and chain offset options will be 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.

For more details about JTAG scan chain settings and download/programming device configurations, see Configuring the JTAG Connection (see page 411).

Table 106: ***FPGA Download Implementation Options***

Option	TCL Option	Description
JTAG Device Name	`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). This implementation option is stored for only this implementation's Flow, and thus does not affect Bitporter pod selection for the Download View (see page 176) or Snapshot Debugger View (see page 253). The pod selection for those views is a user preference, which is managed by the Configure JTAG Connection Preference Page (see page 288), and is not a per-implementation setting. See Configuring the JTAG Connection (see page 411) for more details.

Outline View

The Outline view displays a tree of all pages in the currently active IP Configuration Editor (see page 25). 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 will change as corresponding values are changed in the active IP Configuration Editor. As pages in the Editor are added or removed, entries in the tree will be added/removed accordingly. As values in the Editor page change validity, the validity of the corresponding page in the Outline view's tree will also change.

In addition to showing the pages' validity, the Outline view provides an alternate method for navigating between the various dynamic pages of the IP Configuration Editors (see page 25). Selecting (clicking) an item in the Outline view causes the IP Configuration Editor to turn to the associated page.

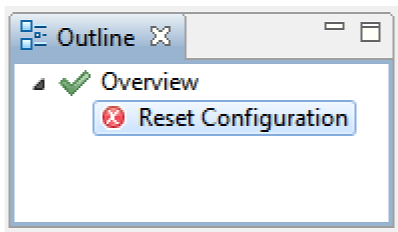


Figure 26: *Outline View Screenshot*

Table 107: *Outline View Icons*

Icon	Description
	No errors or warnings on the page.
	At least one warning on the page, but not errors.
	At least one error on the page.

See also: Creating an IP Configuration (see page 387)

Package View

> **The Package View is only applicable for Speedster FPGA devices**
>
> Users should ignore this View when developing for other Achronix product types.

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

Clicking on 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 23). To add it to the current perspective, select **Window -> Show View -> Other... -> Package**.

See also: Viewing the Package Layout (see page 498) and Pre-Placing a Design (see page 399).

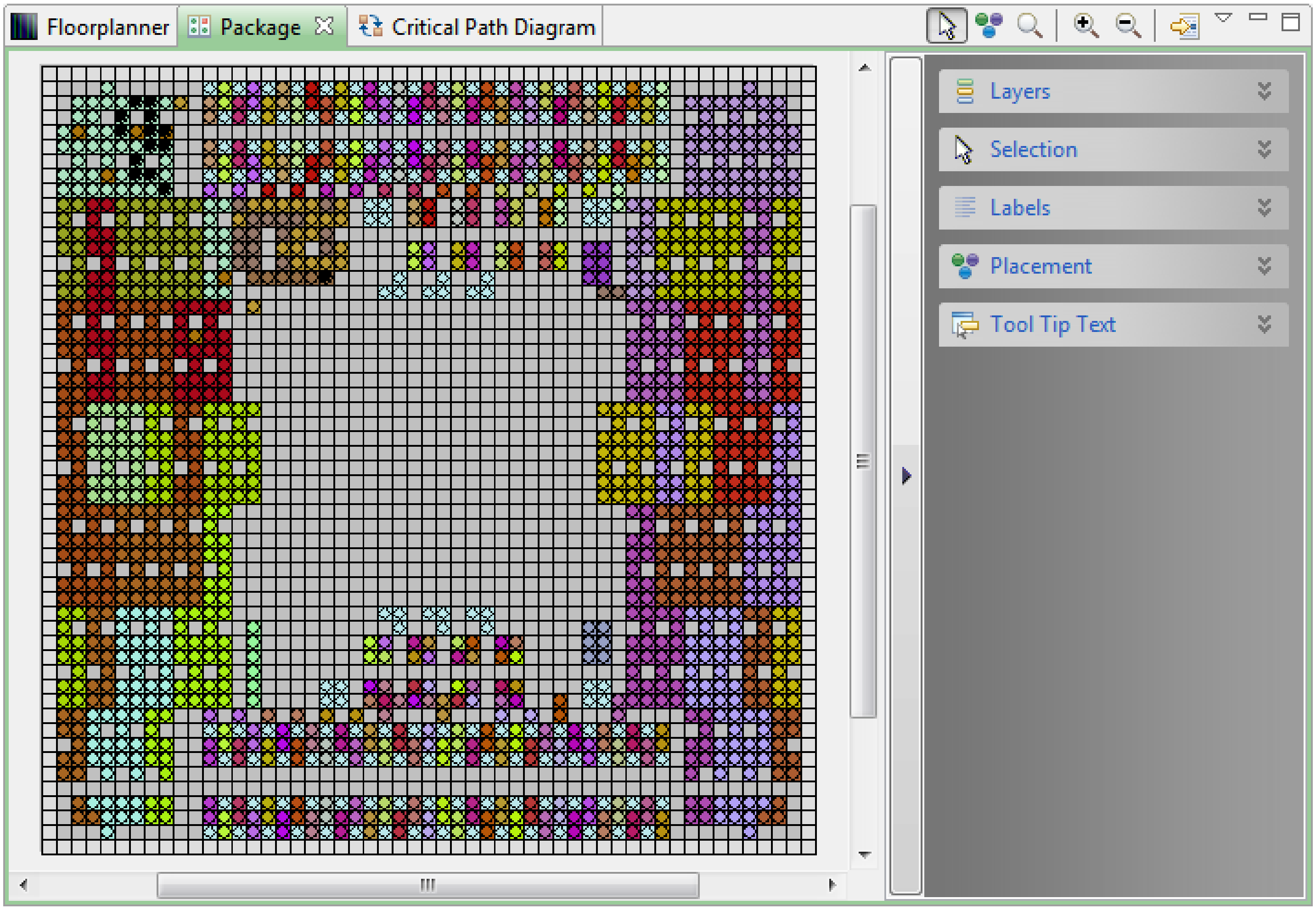


Table 108: *Package View Icons*

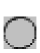
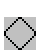
Icon	Type	Description
–	Blank	No package ball at this location
	Circle	Available ball location.
	Diamond	VDD or VSS ball location.
	N/C	Non-connected ball location.

Table 109: ***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 drag when the left mouse button is pressed and held.
	Zoom tool	Controls the behavior of the mouse while in the Package 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 Package view by 200%.
	Zoom out	Decreases the current zoom level in the Package view by 200%.
	Save Placement	Opens the Save Placement dialog allowing the user to save the current placement to a pre-placement constraints file.

Panning and Zooming

The Package view allows the user to zoom in and out, to see more or less details respectively. There are several ways the user may change the zoom level: 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 (see page 498) 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 will 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 will require 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's drag-and-drop interactions. See the task Package Panning (see page 499) 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 a user to filter the view to just the desired objects.

Objects in the current ACE selection set are always visible.

Table 110: ***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 (see page)).

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 249).

Table 111: ***Selection Options***

Option	Default	Description
Action		
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.
Remove Placement	Disabled	This radio button controls the action applied to enabled objects in the selection region. This setting causes the placed instances to be un-placed.

Placement

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

Table 112: *Placement Options*

Option	Default	Description
Fixed Placement	Enabled	This option 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.

> 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 113: *Tooltip Options*

Option	Default	Description
Allow Tooltips	Enabled	Allows users to enable/disable 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 114: *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

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

The Partitions View is used to display the state of the active implementation's partitions, and allows (through interactions with the Floorplanner View (see page 178), Search View (see page 245), and Selection View (see page 249)) 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 443)

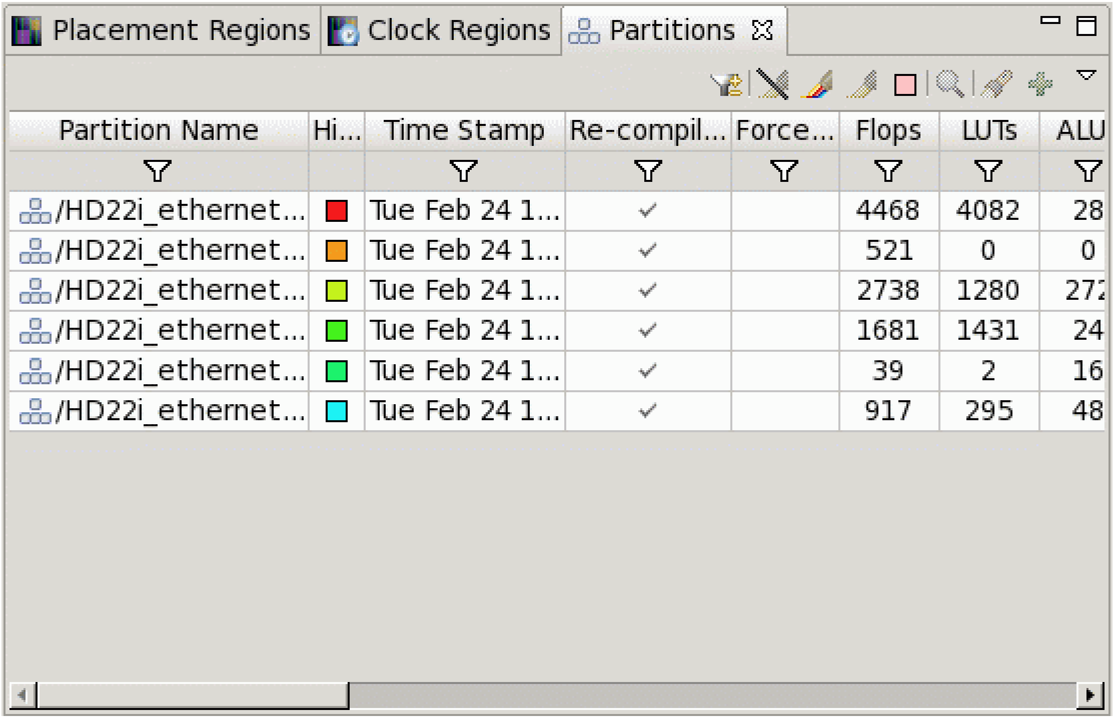


Figure 27: *Screenshot of the Partitions view*

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 by the ACE end user.

Table 115: *Partitions View Columns*

Column	Description
Partition Name	The name of the partition as specified in the design's constraints file(s).
Highlight Color	If all instances within the partition have the same highlight color, the row will show 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 will display no color square.
Time Stamp	The timestamp of the last compile for this partition (compile point) in the upstream synthesis tool.
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 will be forced to re-compile during the next pass through ACE. This checkbox may be toggled on/off by the user 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 116: *Partitions View Actions*

Icon	Action	Toolbar Button	Context Menu	Description
	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 249)).
	Choose Highlight Color	Y		Determines which color will be 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 partition. (See Highlighting Objects in the Floorplanner View (see page 396).)
	Un-Highlight	Y	Y	Clears the Highlight for the instances within the chosen partition.
	Auto-Highlight	Y		Automatically assigns a unique highlight color to each partition.
	Zoom To	Y	Y	Zooms the Floorplanner view and Package View (see page 226) to a region containing the instances within the partition currently chosen in the tree.
	Search for Instances	Y	Y	Searches for instances belonging to the chosen partition. A Tcl `find` command is issued, and the Search View (see page 245) is populated with the results.
	Toggle Filter Row Visibility	Y		Changes whether the filter row (of filter icons) is visible or not. Note that this does not alter whether filters are active, it only changes the visibility of the row of filter icons.
	Force Partition Changed		Y	Override the partition's timestamp during the next pass through Ace: A check mark appears in the **Force Re-compile on Next Run** column, and the partition will be re-placed and re-routed the next time you run the flow, even if there were no RTL changes and it was not re-compiled in the upstream synthesis tool. **Un-Force Partition Changed** will remove the check mark in the column.

Icon	Action	Toolbar Button	Context Menu	Description
	Un-Force Partition Changed		Y	Removes the check mark from the **Force Re-compile on Next Run** column, so the compilation timestamp will no longer be overridden. This is essentially an undo operation for the **Force Partition Changed** action.

Note that 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 23). The view only acts as a Drag-and-Drop source; items dropped on the Partitions view will be ignored.

Any row of the table may be dragged to the Tcl Console view (see page 258), 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 234) or the Floorplanner View (see page 178) (when that view has the **Placement Regions Tool** active) to assign placement region constraints (see page 438) (using the Tcl command `add_region_find_insts`). Dragging a partition is the equivalent of dragging all individual instances which are members of that partition.

Placement Regions View

The Placement Regions view provides a tabular representation of the content of all user-created Placement Regions (see page 435) for the design. The view allows the user to manipulate the visibility of the Placement Region itself as painted (as a colored overlay) in the Floorplanner View (see page 178), and manipulate the content (the instances constrained to the region) of each Placement Region. The view's table will remain empty until the currently active Implementation has completed the **Run Prepare** flow step.

Because users manually define Placement Regions, and users manually constrain Instances to the Placement Regions, users must be able to track the total number of sites and total number of 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 a user ever assigns more Instances 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 435) 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 23). To add the view to another perspective, select **Window → Show View → Other... → Achronix → Placement Regions**.

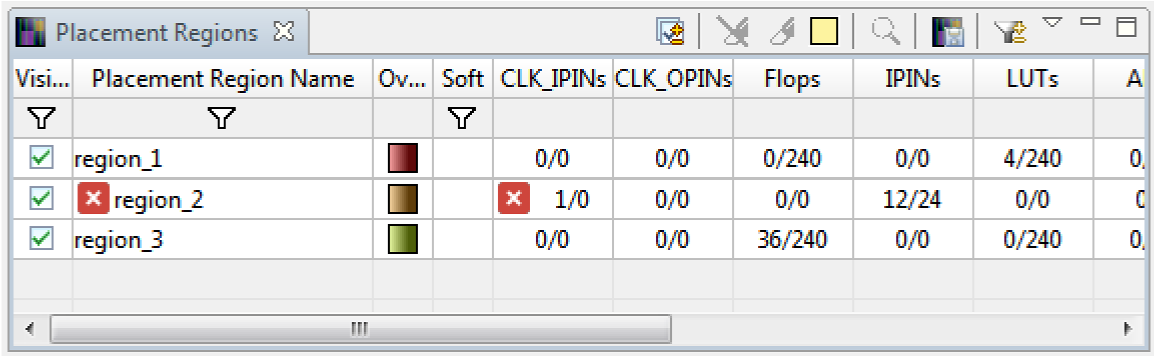


Figure 28: ***Screenshot of the Placement Regions View***

In the above example screenshot, error icons are shown for the CLK_IPINs 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 assist user awareness when an erroneous resource type is scrolled offscreen.

> 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 may not match the exact resources of any particular target device available to the user.

Table 117: ***Placement Regions Table Columns***

Column	Description
Visibility	When selected (this is user-editable), this placement region's overlay will be painted in the Floorplanner View (see page 178), using the chosen translucent **Overlay Color**.
Placement Region Name	The name of this placement region.
Overlay Color	The (user editable) translucent color which will be used to paint an overlay in the Floorplanner View, showing the location of the placement region.
Soft	When the placement region is created, the user can choose to make the placement region a Soft region. Soft regions will contain a checkmark in this column. Users are not allowed to alter whether a placement region is soft at any time other than at creation.
Resource	The number of *resource* Instances constrained to this placement region / The number of *resource* sites contained within the bounds of this placement region

Table 118: *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 178).
	Show / 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 249))
	Deselect Constrained Instances		Y	Removes all Instances constrained within this placement region from the ACE Selection Set.
	Change Overlay Color		Y	Allows the user to choose which translucent Overlay Color will be used to represent this placement region in the Floorplanner View.
	Reset Overlay Color		Y	Reset the chosen placement region's overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two placement regions are too similar for easy discernment, this will pseudo-randomly pick another color. Each time this action is chosen, another color will be 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 will be visible in the Floorplanner view. (See Highlighting Objects in the Floorplanner View (see page 396).)
	Choose Highlight Color for next Highlight command	Y	Y	Allows the user to change the current placement region constrained instances opaque highlight color (which is different from the placement region translucent overlay color). This opaque highlight color will be used in the Floorplanner view when the **Highlight Constrained Instances** () action is chosen in the Placement Regions view.
				Zooms and pans the Floorplanner view to show the location of the selected Placement Region. (The Placement Region itself will not be visible as an

Icon	Action	Toolbar Button	Context Menu	Description
	Zoom to: *region_name*	Y	Y	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. This is done by calling the Tcl command `get_region_insts` (see page 564).
	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.
	Save Placement Regions	Y		Brings up the Save Placement Regions Dialog (see page 284), allowing the user to save one or all 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. Note that 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 178). By default, no Placement Region overlays are painted in the Floorplanner View. Users must choose to enable the Placement Region overlays they wish to have displayed. Users may optionally alter the overlay color for each/all Placement Regions, but these color choices are not persisted between ACE sessions.

> While the user is allowed to choose alternate overlay colors for each Placement Region, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors will be automatically chosen for each Placement Region.

The following are ways to alter the presentation of Placement Region data in the Floorplanner View (see page 178):

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 will be painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view will be redrawn with the chosen Placement Region overlay no longer painted.

Enable/Disable painting of all Placement Regions within the Target Device:

When the user chooses the **Show/Hide All Placement Regions** action, the visibility of all Placement Regions will be 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 by right-clicking the mouse pointer anywhere on the row of the desired Placement Region, then selecting **Choose Overlay Color** from the popup context menu. The user will then be able to use the Color Dialog to choose the desired color for the Placement Region. Note that this is a temporary color change - colors will be reverted to automatically chosen defaults if the user changes the active design, the active implementation, the target device, or closes ACE.

ACE will automatically pick a different overlay color for an individual Placement Region if the user chooses **Reset Overlay Color** from the right-click popup content menu.

Temporarily alter the overlay rendering color for all Placement Regions:

ACE will automatically pick different overlay colors for all Placement Regions if the user chooses the **Reset All Overlay Colors** action.

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 by placing the mouse cursor over the boundary between columns - at this point the mouse cursor should change to indicate resizing is possible. Next, 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 by left-clicking and holding on any column name, then dragging left or right to move the column between any other pair of columns, and release the left mouse button to insert the column header at the new location. While dragging, you will see the dragged column header alongside the mouse cursor, and there will be a thick column header separator showing where the column insertion will occur if the mouse is released at the present cursor location.

Filtering

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

Columns containing text can be filtered by string value (glob string matching by default, but Regular Expression matching is also available, following Java rules, 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 (like the Overlay Color column) will lack a filter icon in the filter row.

To add a filter to a column, the Filter Row must first be visible. (Select the **Toggle Filter Row Visibility** action to show the row if necessary.) Then simply left-click the mouse on the filter icon ( ) for the desired column, which causes a data-appropriate filter dialog to appear. Next, fill in the desired filter values and press **Apply** to apply the filter to the Placement Regions in the table. All values matching that filter will be retained, and all other values will be excluded. Additionally, the background color of the filtered column will change to a bright yellow to indicate the filter is active, and the filter icon at the head of the column will also change to the active filter icon ( ).

To edit (or clear) an existing filter, simply left-click the mouse on the active filter icon ( ), which again causes the data-appropriate filter dialog to appear, this time pre-populated with the current column filter setting. Change the filter value and press **Apply** again to edit the filter, press **Cancel** to leave the filter unchanged, or press **Clear** to remove the filter from the column. If the filter is cleared, the background color of the column will return to the default background color, and the filter icon will also change to the inactive version ( ).

Projects View

The Projects view provides a hierarchical view of the Projects (see page 312) in the Workbench (see page 23). From here, projects can be added and removed, project configurations edited, the active implementation (see page 312) can be chosen, saved, or restored, files may be opened for editing, etc.

Clicking on an implementation activates (see page 316) it. Similarly, clicking on a project activates the first implementation in the project definition. The active project and active implementation will both be displayed in a bold font.

The various Source Files (see page 314) making up a project will be added and removed from this view. Source files of the project are listed in the order in which they will be loaded. To change the order the source files are listed / loaded, see Adding Source Files (see page 355).

By default, the Projects view is included in the Projects perspective (see page 23). 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 351). See also: Project Management Preference Page (see page 306)

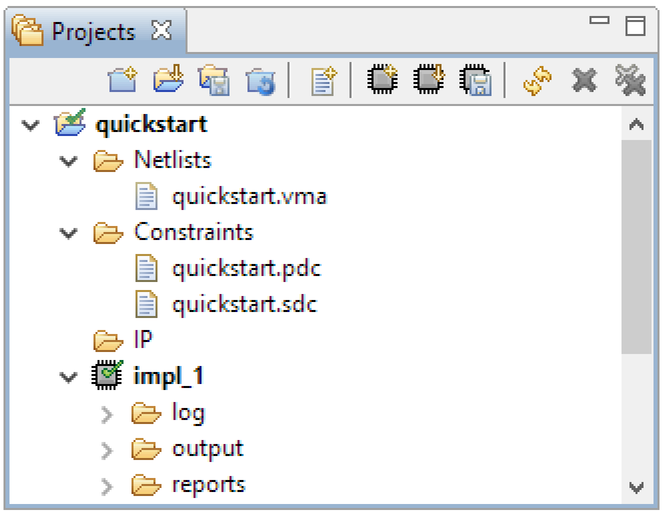


Table 119: *Projects View Actions*

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 273) to allow the user to create a new project definition in the tool. See also: create_project (see page 543)
	Load project	Y		Opens the Load Project Dialog (see page 276) to allow the user to load an existing ACE Project File into the tool. See also: load_project (see page 569)
	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 590)
	Save Project As...		Y	Saves the selected ACE project to a newly-chosen filename/location on disk.

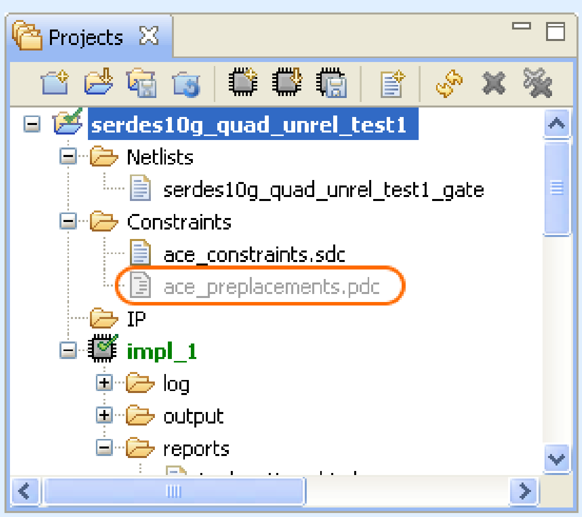
Icon	Action	Toolbar Button	Context Menu	Description
	Reload project	Y	Y	Reloads the selected ACE Project. See also: restore_project (see page 580)
	Add source files	Y	Y	Opens the Add Source Files Dialog (see page 261) to allow the user to add source netlist and constraint files to the selected project in the Projects view. It also allows IP Configuration (see page 387) files (`.acxip`) to be added to the project as a convenience. See also: add_project_ip (see page 535), add_project_netlist (see page 536), add_project_constraints (see page 535) Constraint files are loaded by ACE in the same order they're displayed within this view. For details on how to change this display / load order, see Adding Source Files (see page 355).
	Create implementation	Y	Y	Opens the Create Implementation Dialog (see page 270) to allow the user to create a new project implementation definition for the selected project in the Projects view. See also: create_impl (see page 542)
	Restore implementation	Y	Y	Opens the Restore Implementation Dialog (see page 279) to allow the user to restore the active project implementation from an Acxdb Archive File. See also: restore_impl (see page 579)
	Rename implementation		Y	Allows the user to rename the Implementation. See also: rename_impl (see page 573)
	Save implementation	Y	Y	Opens theSave Implementation Dialog (see page 280) to allow the user to save the active project implementation to an Acxdb Archive File. See also: save_impl (see page 589)
	Open Multiprocess Report		Y	Opens the Multiprocess Summary Report (see page 323) for the selected project, if the project has one. See also: display_file (see page 545)
	Refresh contents	Y	Y	Refreshes the listing of supporting files contained within the selected project or implementation.
	Remove	Y	Y	Allows the user to remove the selected item from the Projects view. Removing items from a project does not delete the corresponding resources on the file system, except for removing implementation output and report files. See also: remove_project (see page 571), remove_project_constraints (see page 571), remove_project_netlist (see page 572), remove_project_ip (see page 572), remove_impl (see page 571)
	Remove all projects	Y		Allows the user to remove all projects from the Projects view. Removing a project does not delete the corresponding resources on the file system.

Icon	Action	Toolbar Button	Context Menu	Description
	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 (see page 554)

Table 120: *Project View Icons*

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

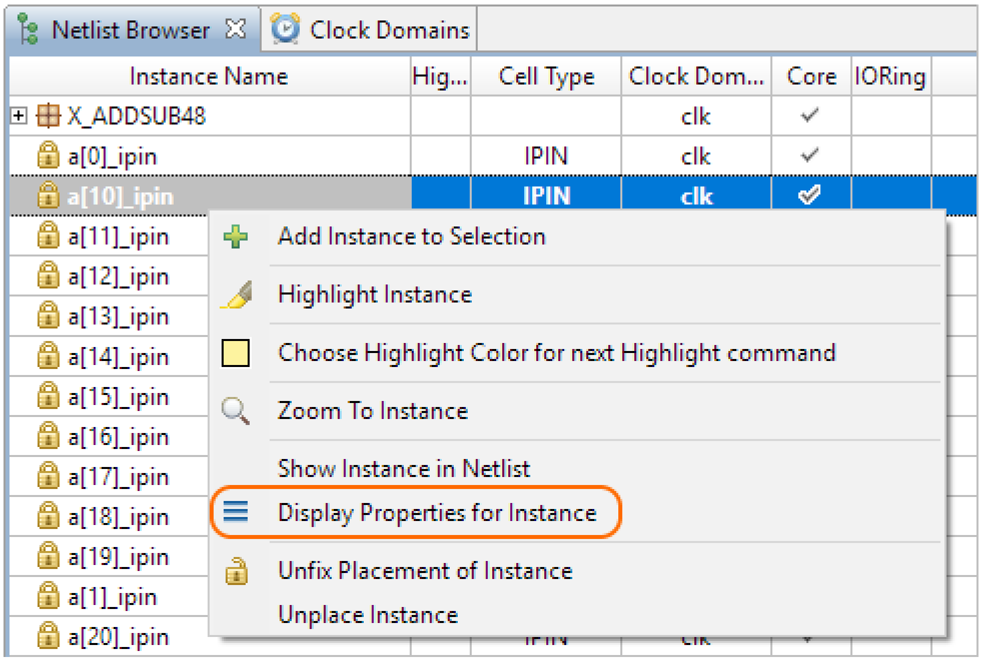
Some files in the 'Constraints' section of the tree may appear greyed-out to indicate that those constraint files are not enabled in the Active Implementation (see page 316). Various constraint files in a project can be enabled or disabled for an implementation in the Options View (see page 215), under **Design Preparation | Constraint Files**.



Properties View

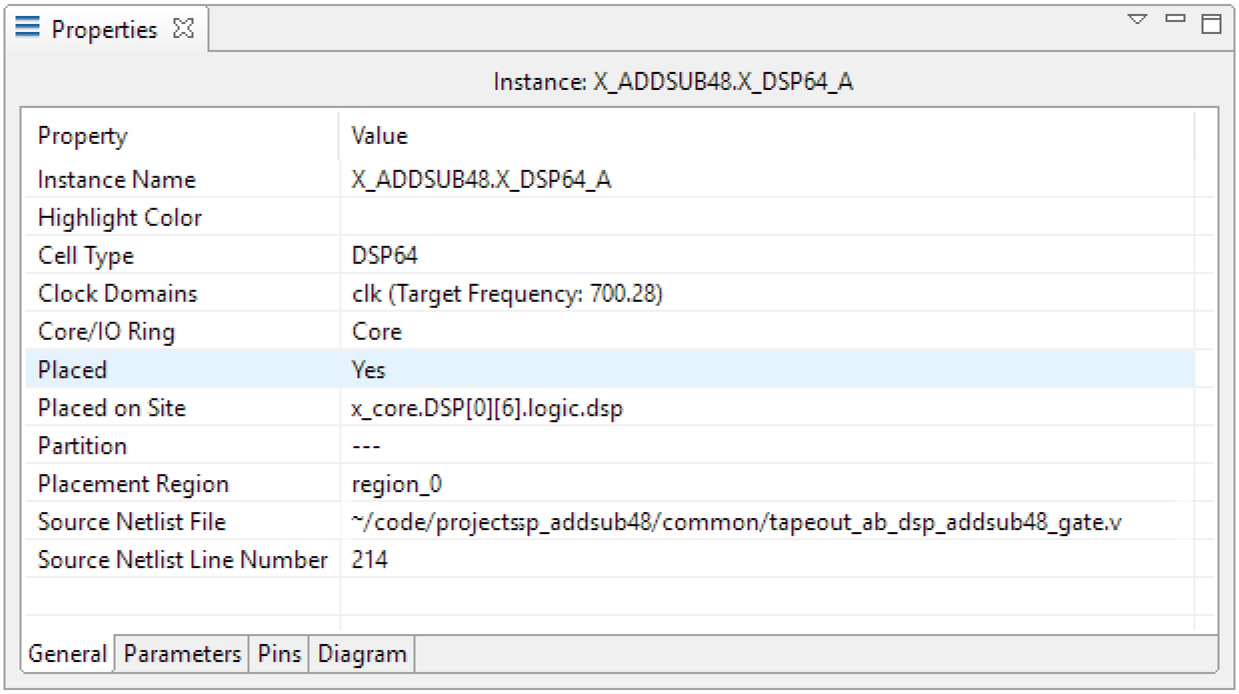
The Properties View can provide in-depth specifics about many types of pin, net, and instance items on demand, and the view then allows users to navigate many of the relationships between connected items.

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 (see page 545) can also be used to populate the Properties View.



General Tab

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

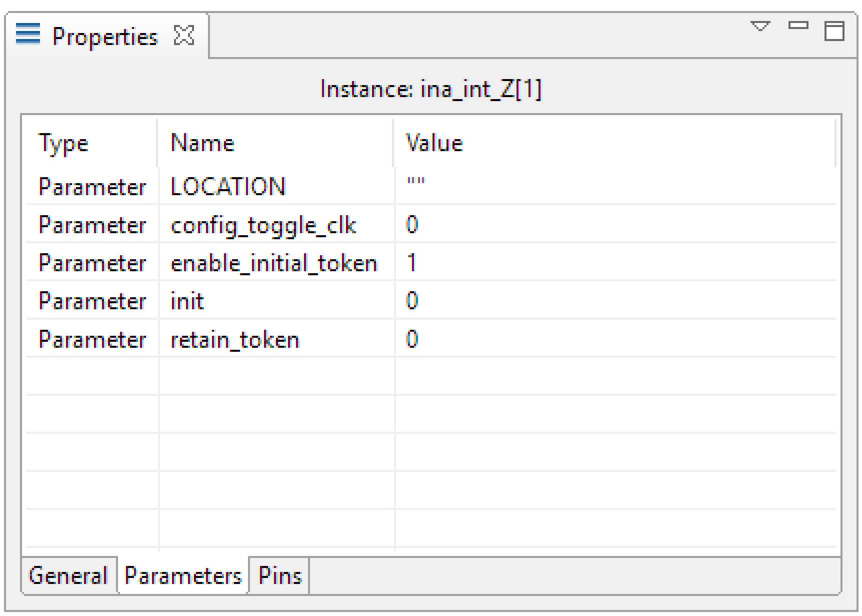


Source Netlist Shortcuts

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

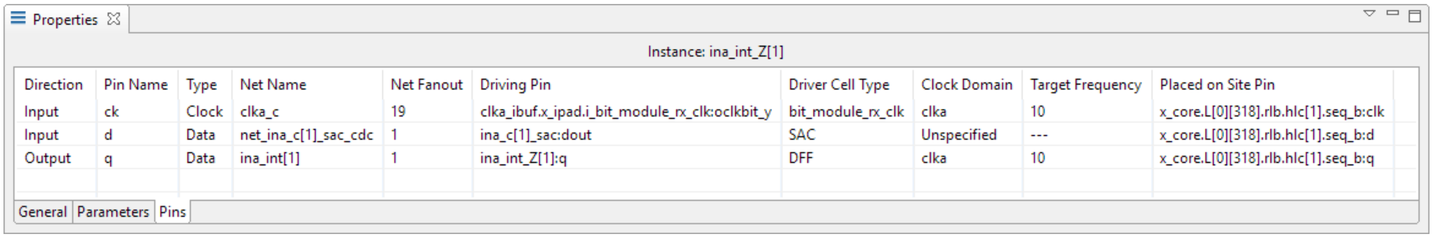
Parameters Tab

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



Pins Tab

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



Direction	Pin Name	Type	Net Name	Net Fanout	Driving Pin	Driver Cell Type	Clock Domain	Target Frequency	Placed on Site Pin
Input	ck	Clock	clka_c	19	clka_ibuf.x_ipad.i_bit_module_rx_clk:oclkbit_y	bit_module_rx_clk	clka	10	x_core.L[0][318].rlb.hlc[1].seq_b:clk
Input	d	Data	net_ina_c[1]_sac_cdc	1	ina_c[1]_sac:dout	SAC	Unspecified	---	x_core.L[0][318].rlb.hlc[1].seq_b:d
Output	q	Data	ina_int[1]	1	ina_int_Z[1]:q	DFF	clka	10	x_core.L[0][318].rlb.hlc[1].seq_b:q

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

Table 121: *Properties View Pins Tab Actions*

Icon	Action	Advanced	Description
	Copy Cell Text		Copy 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 249)).
	Remove from Selection		Removes the item from the ACE selection set (as shown in the Selection View (see page 249)).
	Highlight		Applies the currently active highlight color to the chosen item (see Highlighting Objects in the Floorplanner View (see page 396)).
	Choose Highlight Color		Determines which color will be applied the next time the Highlight action is selected.
	Zoom To		Zooms the Floorplanner view and Package view to a region containing the item.
	Show in Netlist		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). **Caution!** This is Early Access functionality; this may not always open the text editor to the expected location.
	Display Properties		Display properties for the chosen item in the Properties view. (See Properties View (see page 242))

Reminder: Instances' Selection color vs Highlight color priority

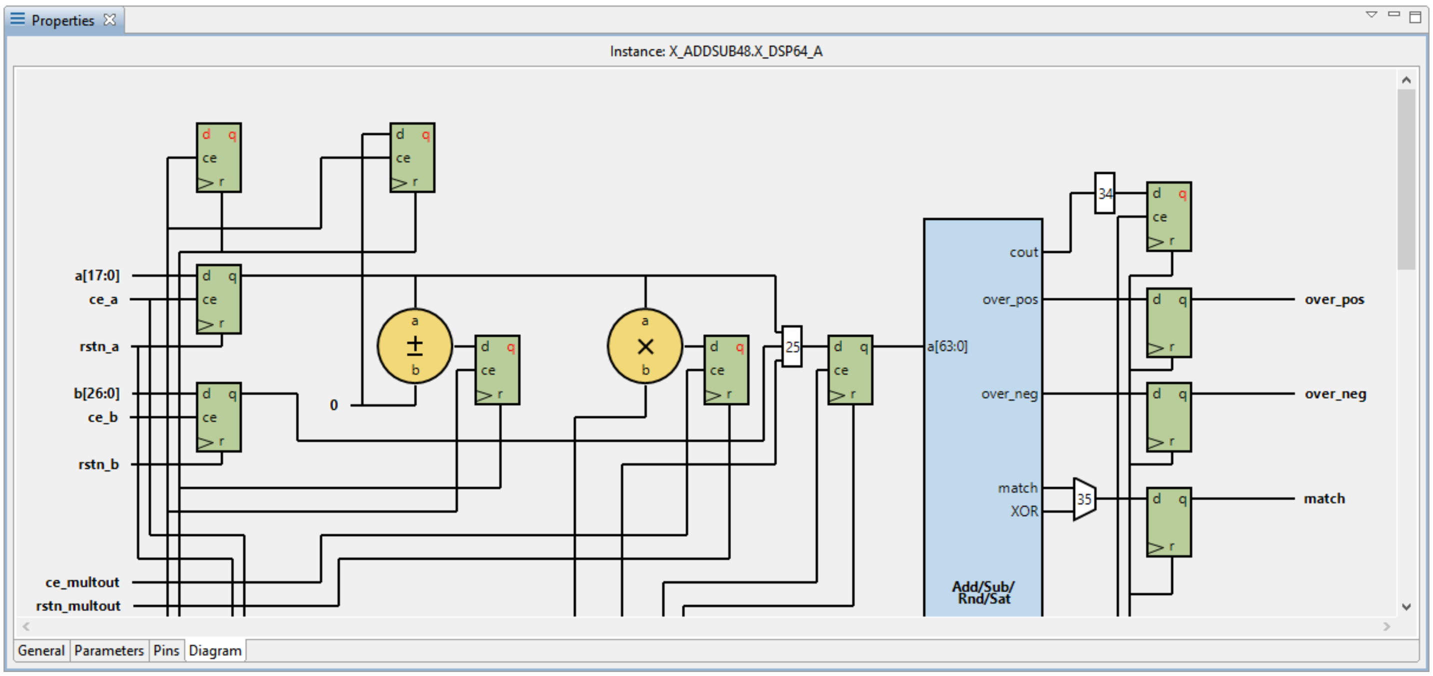
With default preference settings, in the Floorplanner View (see page 178), highlight colors of (placed) instances are only visible when the Instances Layer is enabled, and the instances are not members of the ACE selection set. This behavior is due to the instance's selection color has a higher priority 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 may provide supplemental information where useful.



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 249). The Search View is a graphical interface to the Tcl command `find` .

Instances and Ports in the results list may be dragged and dropped onto the Floorplanner View (see page 178) to assign placement or add placement region (see page 435) constraints (the behavior depends upon the Floorplanner's active tool /mode). Ports may also be dragged and dropped onto the Package view (see page 226) to assign placement. Instances and Paths in the results list may be dragged to the Placement Regions View (see page 234) to add placement region constraints.

By default, the Search view is included in the Floorplanner perspective (see page 23). To add it to the current perspective, select **Window -> Show View... -> Search**.

See also: Object Type Prefixes (see page 385)

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

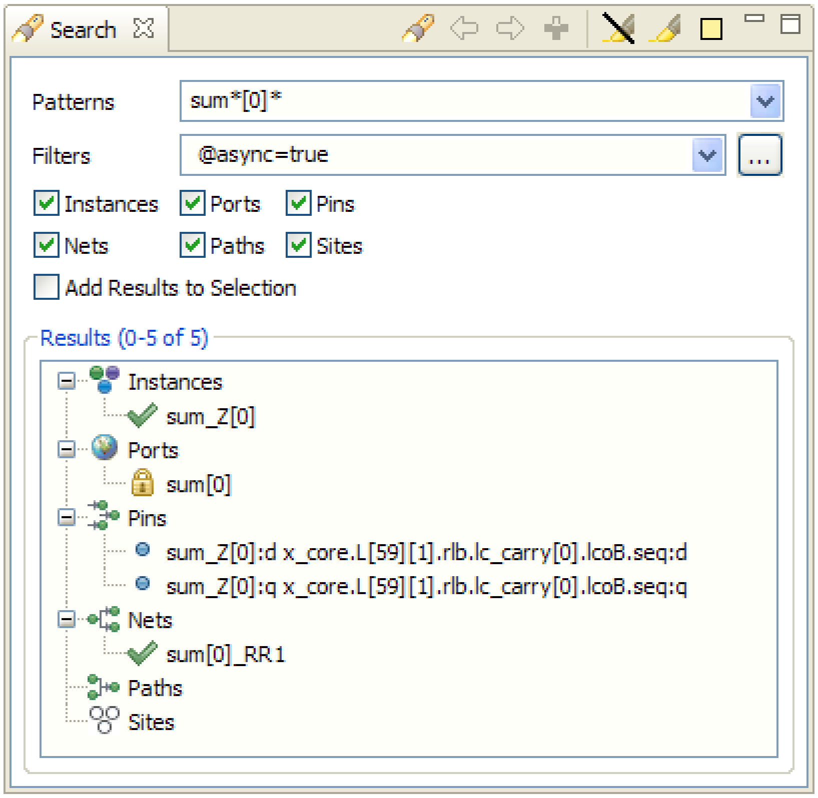
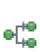


Table 122: *Search View Icons*

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 will be under this branch of the search results.)
	Ports (All ports will be under this branch of the search results.)
	Pins (All pins will be under this branch of the search results.)
	Nets (All nets will be under this branch of the search results.)
	Paths (All paths will be under this branch of the search results.)

Icon	Description
	Sites (All sites will be 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 will only affect the subset of items currently chosen within the Results list.

Table 123: *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	Y	Y	Turns off the highlight color for objects. (Note: Stops highlighting the search results in the Floorplanner View. Other views are not affected by highlighting.)
	Highlight Results	Y	Y	Highlights objects with the currently-selected search highlight color. The highlighted results will be visible in the Floorplanner View. (Other views are not affected by highlights.)
	Choose Highlight Color	Y		Allows the user to change the current highlight color for search result highlighting. This color will be used in the Floorplanner View when the Search view's **Highlight Results** ( ) button is pressed.
	Zoom To Object		Y	Zooms the Floorplanner view to a region containing the item currently chosen in the results list.
				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.)

Icon	Action	Toolbar Button	Context Menu	Description
	Show in Netlist		Y	This is Early Access functionality; this may not always open the text editor to the expected location.
	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).
	Unplace Instance (s)		Y	Unplaces all Instances currently chosen in the results list.

Table 124: *Search View Options*

Option	Description
Patterns	Enter a Tcl regular-expression pattern which will be used to perform a name-based search. Previously used search patterns may be selected from the drop-down.
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. See Filter Properties (see page 329).
”...” (Search Filter Builder)	This button opens the Search Filter Builder Dialog (see page 285) to guide the user through the options available for search filters.
Instances	Select this checkbox to include Instances in the search results.
Ports	Select this checkbox to include Ports in the search results.
Pins	Select this checkbox to include Pins in the search results.
Nets	Select this checkbox to include Nets in the search results.
Paths	Select this checkbox to include Paths in the search results.
Sites	Select this checkbox to include Sites in the search results.
Add Results to Selection	If selected when a search is performed, all the results of that search will be added to the ACE selection set.

Option	Description
	Caution! If none of the object-type option checkboxes are checked, the search will be 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 greem, 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, a single entry in the “Results” list can be double-clicked to add it to the current selection.

Search Highlights

There is typically a tremendous amount of visualization data available in the Floorplanner view (and to a lesser extent the Package 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 396)) is an attempt to help turn this data into useful information, by enabling the user to find and focus on specific information within their designs.

By highlighting multiple search result sets in the same or different colors, the user can make desired information 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), the user may focus in on just the objects that most interest them.

When used in combination with the layering functionality (see the Layers portion of the toolbox for the Floorplanner view and Package view) and Selection functionality (see the Selection view (see page 249) as well as the Tcl commands `select` and `deselect`), the user should be able to achieve a graphical visualization at whatever granularity they desire.

> The Selection color takes precedence over the Highlight color by default. If design objects are both highlighted and selected, they will be painted the selection color (bright green by default) in the Floorplanner view and Package 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's settings (including precedence) for Highlight and Selection colors can be manipulated on the Floorplanner View Colors and Layers Preference Page (see page 292).

Selection View

The Selection view provides an interface allowing a user to view and manage 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 Tcl commands `select` and `deselect` .

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.

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

The (current page of) selected objects in the Selection view will also be displayed with special coloration (by default a bright green) in the Floorplanner view (see page 178) and the Package view (see page 226).

Objects may be added to the selection set from the Search view (see page 245) (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**), from the Floorplanner view (see Selecting Floorplanner Objects (see page 394)), or from the Package view (see Selecting Package Objects (see page 499)).

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 the user wishes to drag all selected objects of a given type (for example, Instances), including those not in the current page of 200 selected objects, then the node with that type name (ex: Instances) may be dragged. (Be aware that some drag-and-drop operations, like pre-placement assignment, will not work with multiple selected objects at once.)

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 435) constraints (the behavior depends upon the Floorplanner's active tool /mode). Ports in the selection list may be dragged and dropped onto the Package view to assign pre-placement. Instances and Paths in the results list may be dragged to the Placement Regions view (see page 234) to add placement region constraints.

By default, the Selection view is included in the Floorplanner perspective (see page 23). To add it to the current perspective, select **Window -> Show View... -> Other... -> Selection**.

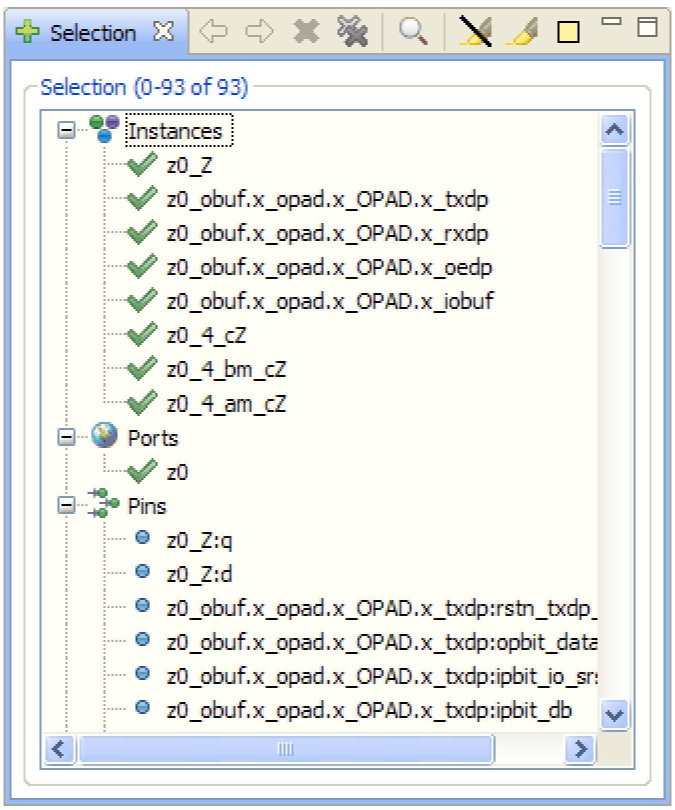


Table 125: *Selection View Icons*

Icon	Description
	Object

Icon	Description
	Placed Object (Applies to instances and ports)
	Fixed-Placed Object (Applies to instances and ports)
	I/O Macro (Applies to ports)
	Instances (All instances will be under this branch of the selection.)
	Ports (All ports will be under this branch of the selection.)
	Pins (All pins will be under this branch of the selection.)
	Nets (All nets will be under this branch of the selection.)
	Paths (All paths will be under this branch of the selection.)
	Sites (All sites will be 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 will only affect the subset of items currently chosen within the Selection list. Be aware that available right-click context menu choices will vary depending upon the context: the number and the type of the items will alter the available actions.

Table 126: *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	Y	Y	Turns off the highlight color for objects in the current selection. (Note: Stops highlighting the ACE selection set in the Floorplanner view. Other views are not affected by highlighting.)
	Highlight Selection	Y	Y	Sets the highlight color for objects in the current ACE selection set to the currently-chosen highlight color. Reminder: the highlight coloring will only be 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 the user to change the current ACE selection set highlight color (which is different from and overridden by the Selection color). This color will be used in the Floorplanner view when the **Highlight Selection** ( ) action is chosen in the Selection view.
	Show in Netlist		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.) This is Early Access functionality; this may not always open the text editor to the expected location.
	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.

For more information about the interaction between Selection and Highlighting, please see Search Highlights (see page) as well as Highlighting Objects in the Floorplanner View (see page 396).

See also: Object Type Prefixes (see page 385)

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 23). To access the Snapshot Debugger view from any other perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Snapshot Debugger.**

From this view, the user is able to run the Snapshot Debugger (see page 415) embedded in their design. A simple button press will Collect Live Sample Data (see page 427) in a VCD file. In addition, in this view the user can Configure the Debug Capture Trigger Pattern(s) (see page 419), Configure a Test Stimulus (see page 424), and Configure the Data Capture Ranges (see page 426) before and after the trigger(s).

For convenience, the user may choose to save (see page 429) ( ) and load (see page 429) ( ) favorite Snapshot configurations via the view's toolbar buttons. Saved configurations may also be used to drive Snapshot in Batch Mode (see page 430) via Tcl.

> When a user design containing the `ACX_SNAPSHOT` macro completes the Flow Step (see page 317) **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 316) is available, the Snapshot View automatically loads the `names.snapshot` file to pre-populate the relevant fields of the view. Note that 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 430) via Tcl.

See also: Running the Snapshot Debugger (see page 415), Assign Bussed Values Dialog (see page 264), and Assign Bussed Signal Names Dialog (see page 262).

Table 127: *Snapshot Debugger View Toolbar Buttons*

Icon	Action	Description
	Arm Snapshot	Send the trigger conditions configuration to the Snapshot Debugger core, send the *Stimulus* value to the Design-Under-Test, wait for the trigger condition to be met, retrieve the trace buffer contents, and output 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 429).
	Load Snapshot Configuration	Loads a previously saved configuration file. See Saving/Loading Snapshot Configurations (see page 429).
	Capture Snapshot Startup Trigger	The action requires that the end user has configured the initial startup trigger parameters 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. This action will wait for the startup trigger condition to be met, retrieve the trace buffer contents, and output a VCD file.

Icon	Action	Description
	Configure JTAG Interface	Opens the preferences dialog with the Configure JTAG Connection Preference Page (see page 288) visible. See Configuring the JTAG Connection (see page 411).

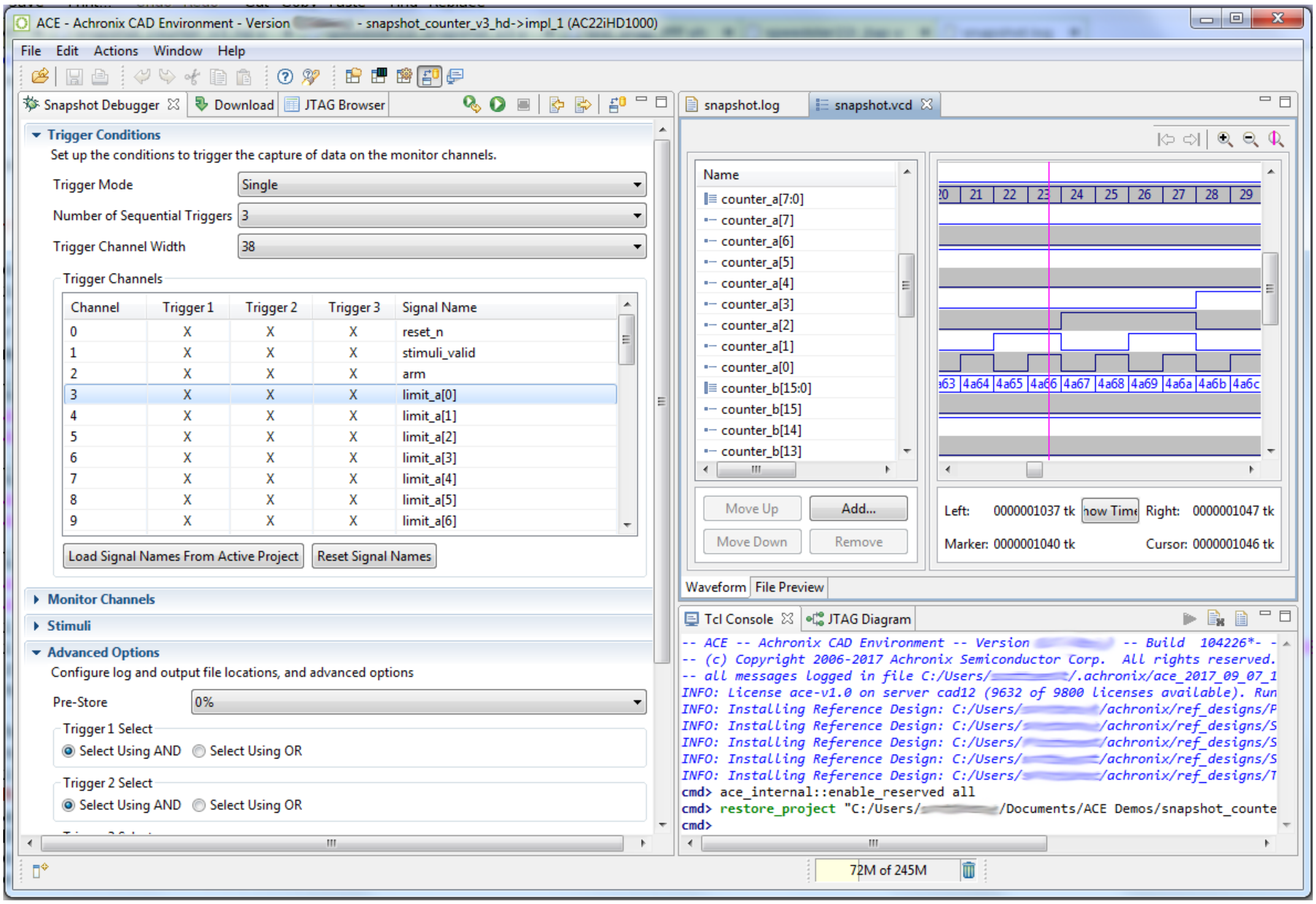


Table 128: *Snapshot Debugger View Options*

Option	Description
Trigger Conditions	
Trigger Mode	Allows the user to select the trigger mode to use when the Arm action is run. 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. 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. 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.
Number of Sequential Triggers	Allows the user to select the use of either **1**, **2**, or **3** sequential triggers. If **1** is selected, Trigger 2 and Trigger 3 are ignored during the match. If **2** is selected, Trigger 3 is ignored during the match 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 by Trigger 2, followed by Trigger 3. See Configuring the Trigger Pattern (see page 419), Configuring Test Stimulus (see page 424), and Configuring the Monitor Signals (see page 423).
Trigger Channel Width	The Snapshot debugger module is parameterizable to trigger channel widths of **1** to **40** channels. The user must set the Trigger Channel Width to the value that corresponds with the parameterized 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), and 1 (level 1). See Configuring the Trigger Pattern (see page 419)
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), and 1 (level 1). This column is only editable if 2 or 3 triggers are selected. See Configuring the Trigger Pattern (see page 419)
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), and 1 (level 1). This column is only editable if 3 triggers are selected. See Configuring the Trigger Pattern (see page 419)
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	

Option	Description
From Active Project	When pressed, 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 pressed, renames all signals back to their default names, which will be "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 will repeatedly Arm the Snapshot debugger and capture samples. The user may set this to automatically run Snapshot up to 128 times.
VCD Record Limit	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.
Overwrite 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.
Monitor Channels	
Monitor Channel Width	The Snapshot debugger module is parameterizable to monitor channel widths of 1 to 4087 channels. The user must set the Monitor Channel Width 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 is parameterizable to capture between 512 and 16384 samples. The user must set the Number of Samples 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 will be used in the VCD file waveform output.
Load Signal Names	

Option	Description
From Active Project	When pressed, 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 pressed, renames all signals back to their default names, which will be "signal" with a suffix corresponding to the channel number.
Stimuli	
Stimuli Channel Width	The Snapshot debugger module is parameterizable to stimuli channel widths of 0 (no stimuli) to 512 channels. The user must set the Stimuli Channel Width 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 426).
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 will be generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 419).
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 will be generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 419).
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 will be generated if **ANY** of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (see page 419).
Frequency (MHz)	Must be configured to match the the *user_clk* timing constraint set in the SDC file of the design being debugged. This will automatically be set according to the values captured in the `names.snapshot` file when an active implementation is available. See Configuring Advanced Options (see page 426).

Option	Description
File Paths Relative To	Chooses whether the **Log File** and **Waveform File** paths are understood to be relative to the **Active Project**'s directory or to the **Working Directory**. (Only matters when the file paths provided are relative paths, and 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/file. See Configuring Advanced Options (see page 426).
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/file. See Configuring Advanced Options (see page 426).
Startup Trigger	This is the same as the **Capture Snapshot Startup Trigger** ( ) button in the view's toolbar. See Collecting Samples of the User Design (see page 427).
Arm	This is the same as the **Arm Snapshot** ( ) button in the view's toolbar. See Collecting Samples of the User Design (see page 427).
Cancel	This is the same as the **Cancel Snapshot** ( ) button in the view's toolbar. See Collecting Samples of the User Design (see page 427).

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, if the user wishes to edit and re-issue 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 user's typing when TAB is pressed, a beep error tone will be sounded, and no content-assist list will appear.) Pressing the up or down arrows on the keyboard will move through entries in the content-assist list, and pressing **Enter** will choose the selected entry in the list. Entries in the list may also be left-clicked with the mouse.

By default, the Tcl Console view is included in all Perspectives (see page 23). 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 382), check the available preferences on the Tcl Console View Preference Page (see page 307), and see the available commands in the Tcl Command Reference (see page 510).

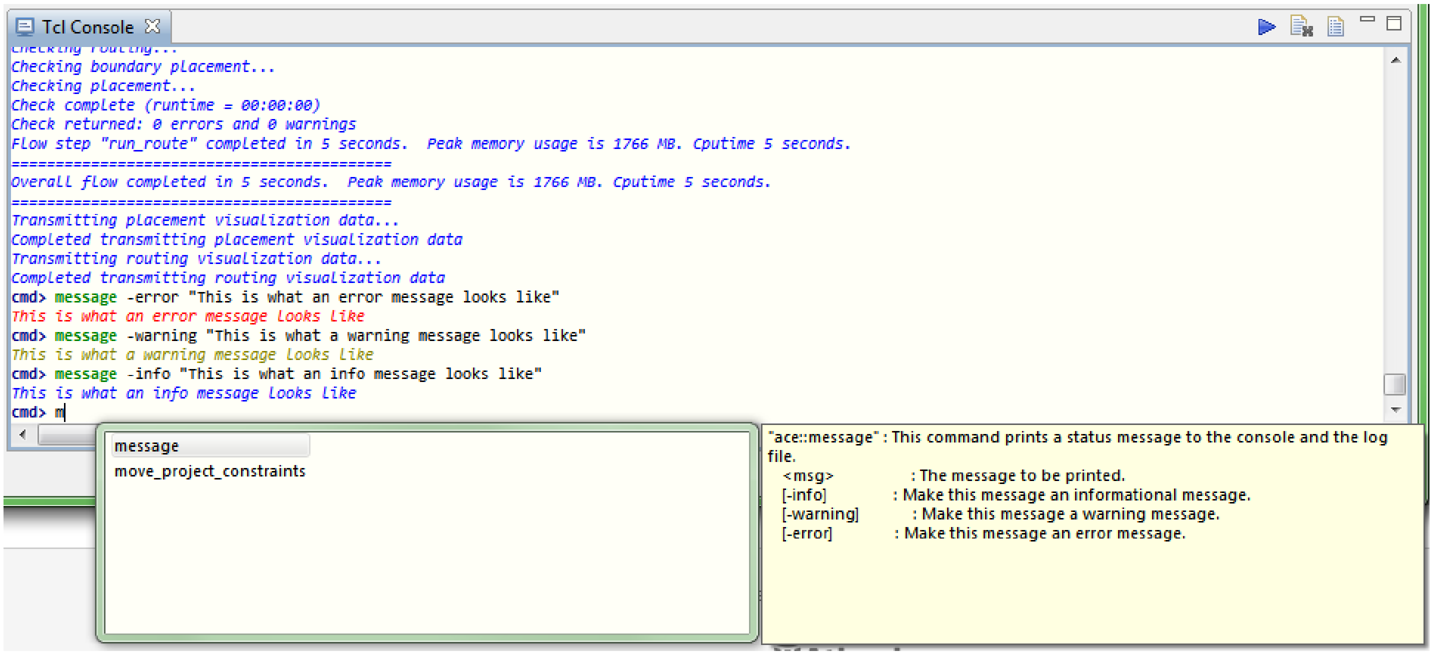


Figure 29: ***Screenshot of the Tcl Console View***

Table 129: ***Tcl Console View Toolbar Buttons***

Icon	Action	Description
	Send command	Sends the current Tcl command at the console's prompt. Alternatively, the user can press **ENTER** 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.

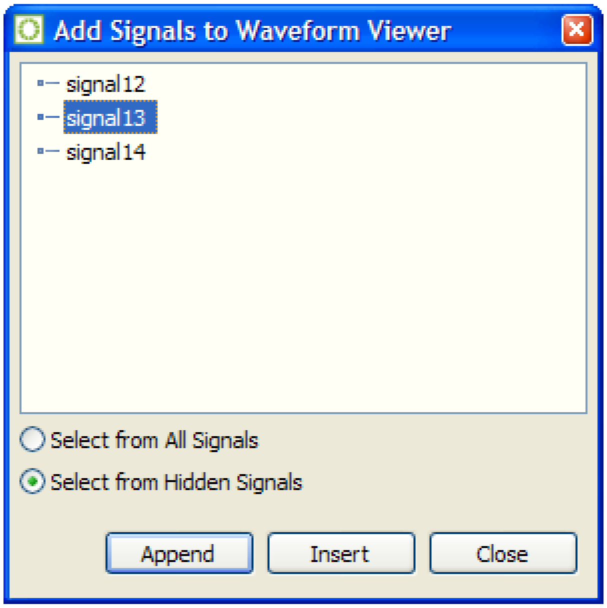
When Tcl command return values are displayed in the Tcl Console, any long returned values will be visually truncated at 500 characters in the console. The actual returned value will not be edited, just the textual representation shown in the console, thus scripts using long return values will still behave 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

This dialog allows the user to add signals to the table and waveform area of the VCD Waveform Editor (see page 27). The dialog allows the user to either unhide signals which were previously hidden via the **Remove** button in the VCD Waveform editor, or add duplicates of already-shown signals to the table and waveform. The selected signals may added at either the top or bottom of the table/waveform via the dialog. If the user wishes to move the signals to a location other than the top or bottom, this must be done via the **Move Up** and **Move Down** buttons on the VCD Waveform editor.



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

Table 130: *Add Signals to Waveform Viewer Dialog Actions*

Action	Description
Select from All Signals	When selected, this radio button will cause the list to be populated with all the signals contained in the current VCD file.
Select from Hidden Signals	When selected, this radio button will cause the list to be populated with all the signals found in the VCD file which are currently hidden. (Signals 'removed' from the VCD Waveform editor's signal table are considered hidden.) If no signals are currently hidden, the list of hidden signals will be empty.
Append	This button appends the currently-selected signal to the bottom of the VCD Waveform Editor's signal table.
Insert	This button inserts the signal currently selected in the dialog's list below the signal currently selected in the VCD Waveform Editor's signal table. If no signal was selected in the VCD Waveform Editor's 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's signal table.
Close	This button closes the dialog.

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

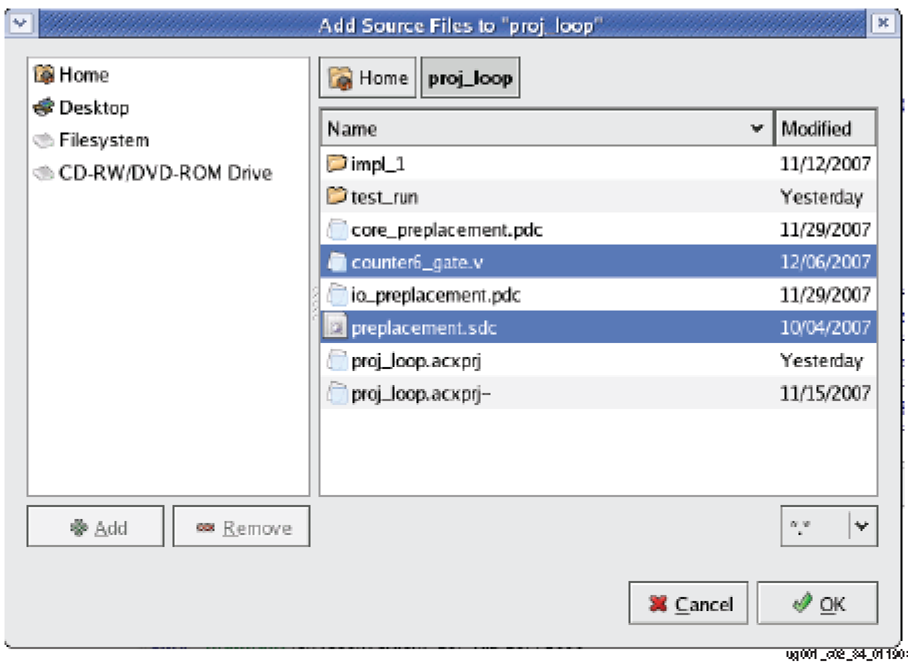
There are also some icons used by content displayed in the dialog's signal list. These are shown below.

Table 131: *Add Signals to Waveform Viewer Dialog Icons*

Icon	Description
	Signal
	Bus

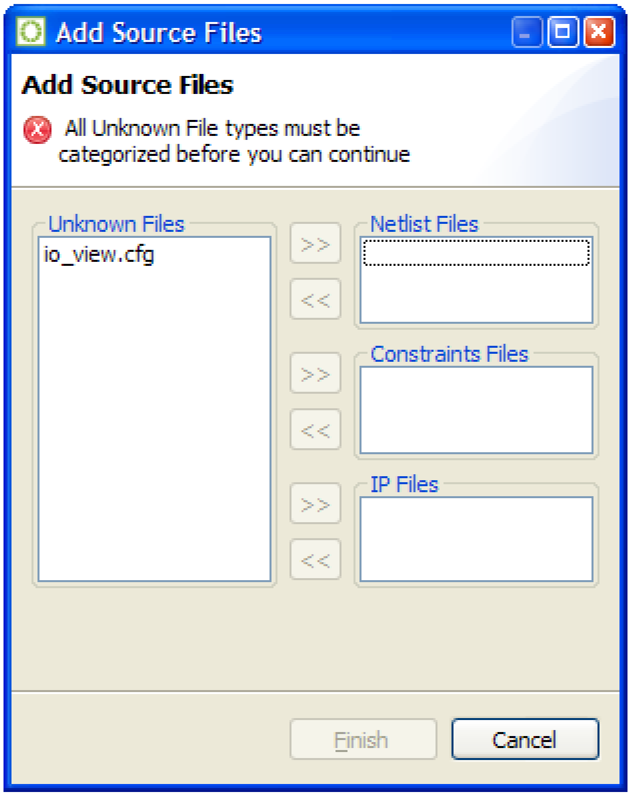
Add Source Files Dialog

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 314) to add to the selected Project (see page 312). After selecting the files to add, click **Open** (in Windows) or **OK** (in Linux) to add them to the project.



> **Source File Load Order**
>
> ACE loads source files in the same order they were added to the project. If ACE is loading files in an incorrect order, remove all source files from the project, and then add them, ***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 will appear asking the user to categorize the unknown file extensions.



The categorization dialog will contain 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.

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

See also: Adding Source Files (see page 355) and Adding Configuration Files to a Project (see page 389).

Assign Bussed Signal Names Dialog

The Assign Bussed Signal Names Dialog wizard allows the user to assign multiple signal names from the SnapShot Debugger view's (see page 253) "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 will then be used in the SnapShot sampled output, visible in the VCD Waveform Editor (see page 27).

> 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.

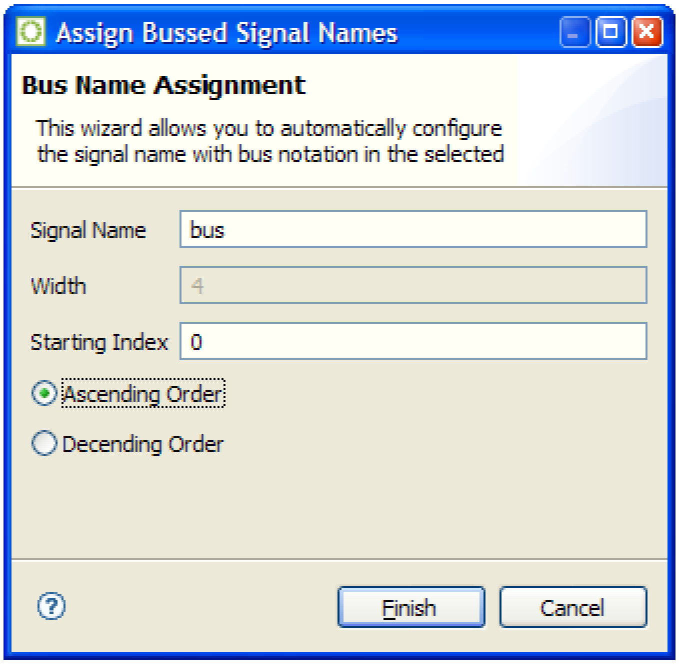


Table 132: *Assign Bussed Signal Names Dialog Options*

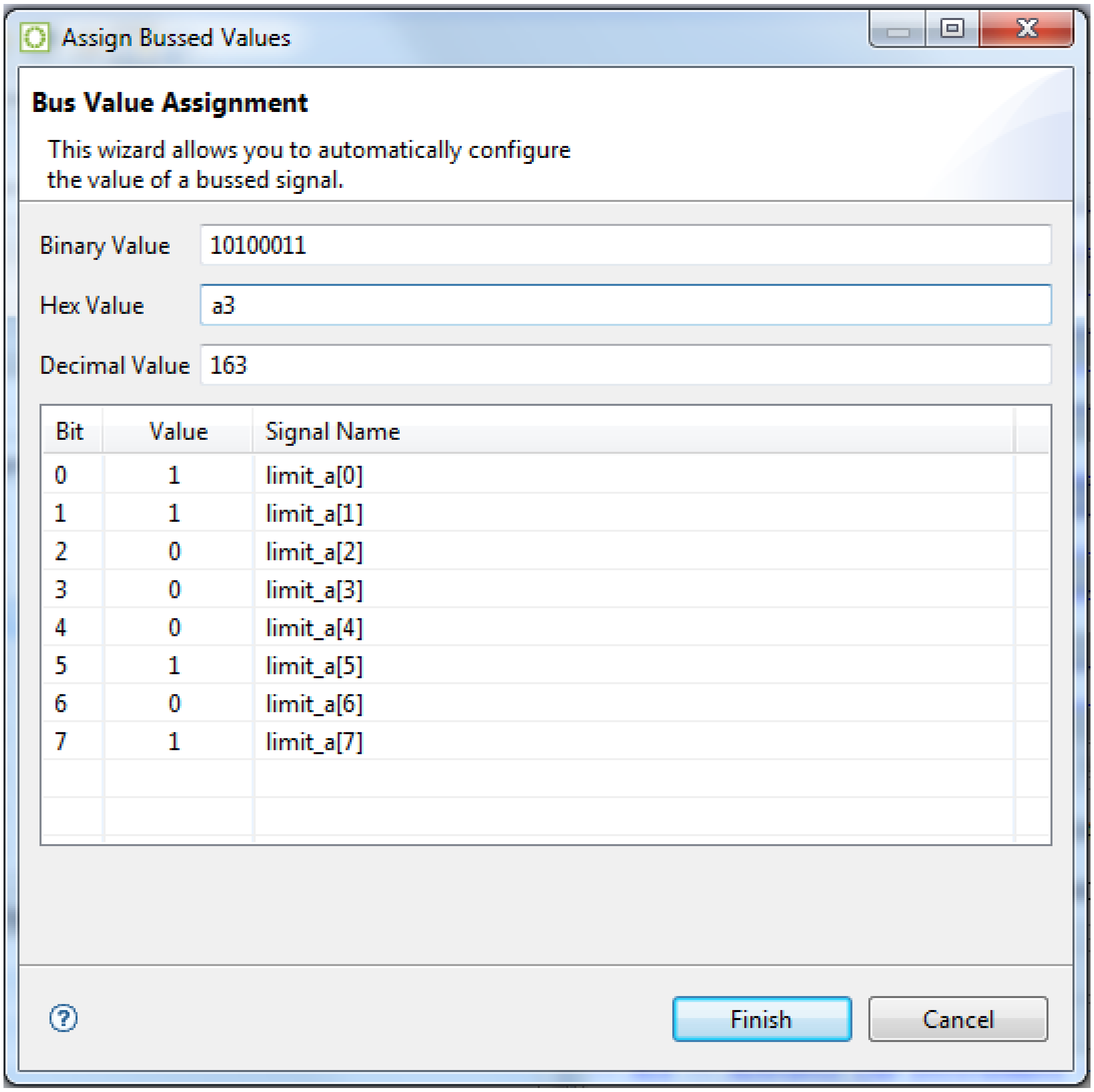
Option	Description
Signal Name	The desired name of the bus.
Width	The width (in bits) of the bus. This is not editable. It will reflect the number of signals which were selected from the table in the Snapshot Debugger View (see page 253).
Starting Index	The desired starting index of the bus, sometimes also called the offset into the bus.
Ascending Order	When selected, the bus indices will start at **Starting Index** and will increment **Width** times.
Descending Order	When selected, the bus indices will start at **Starting Index** and will decrement **Width** times.
Finish	This button will accept the specified bus configuration, close the dialog, and apply the changes to the SnapShot Debugger view's (see page 253) table.
Cancel	This button will discard the specified bus configuration and close the dialog. No changes will be applied to the SnapShot Debugger view's (see page 253) table.

Assign Bussed Values Dialog

The Assign Bussed Values Dialog wizard allows the user to assign a value to multiple signals from the SnapShot Debugger view's (see page 253) "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 253). There are two ways to launch this dialog to allow bus assignment of values:

- With your mouse, left click to select a single row in the SnapShot Debugger View (see page 253) 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.
- With your mouse, hold CTRL or SHIFT and left click to select multiple rows in the SnapShot Debugger View (see page 253) 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.

See also: Configuring the Trigger Pattern (see page 419).



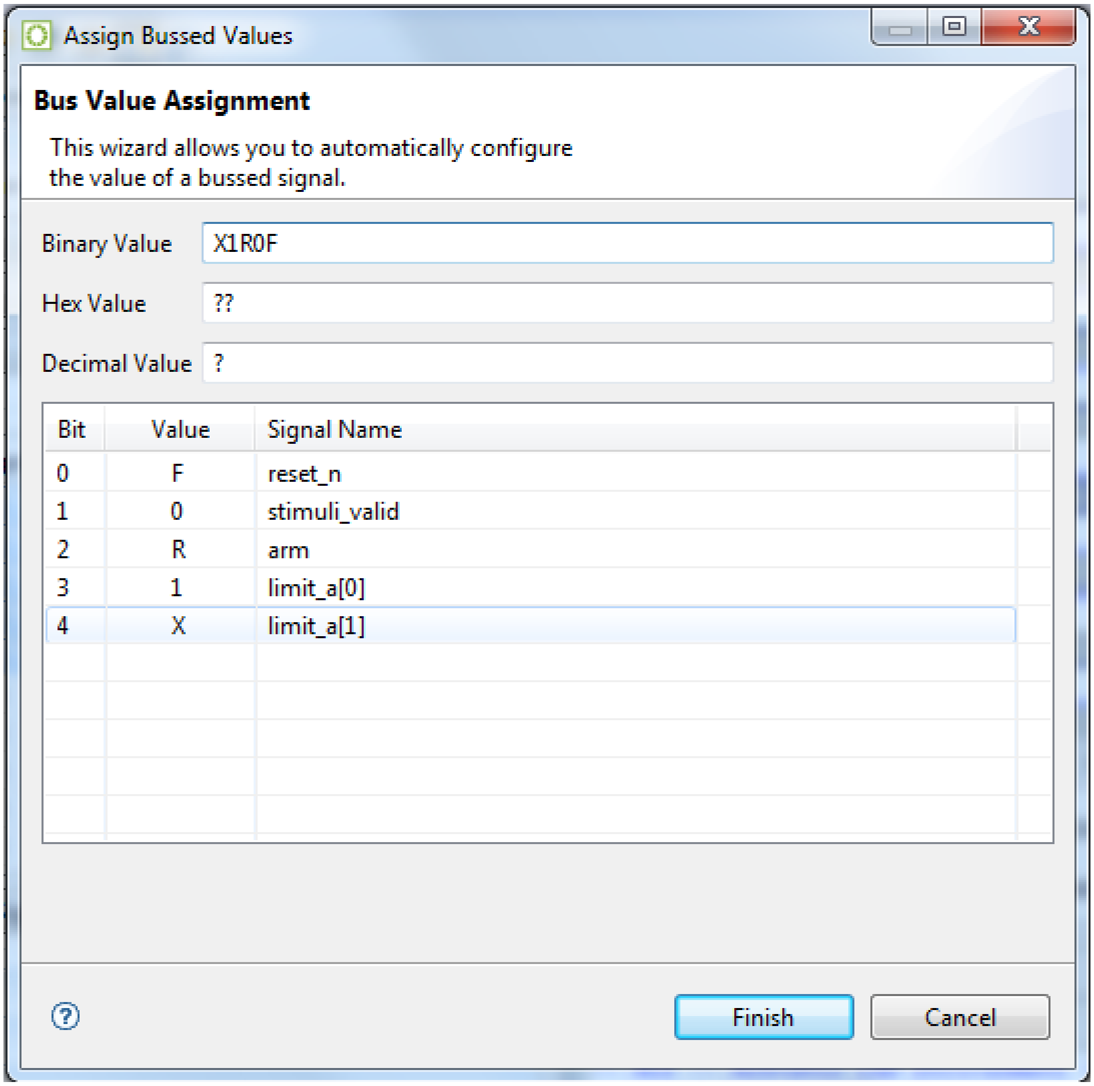


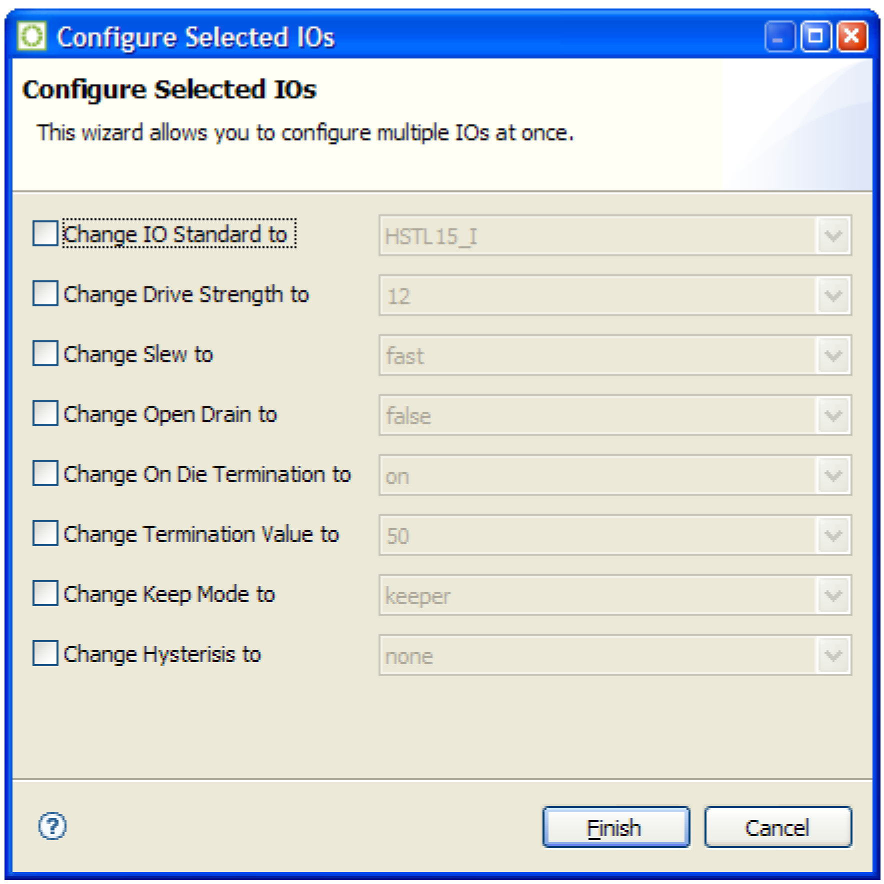
Table 133: *Assign Bussed Values Dialog Options*

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.
Bit	The bit offset into the bus value being edited.

Option	Description
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	This button will accept the specified bus configuration, close the dialog, and apply the changes to the corresponding SnapShot Debugger view's (see page 253) table.
Cancel	This button will discard the specified bus configuration and close the dialog. No changes will be applied to the corresponding SnapShot Debugger view's (see page 253) table.

Configure Selected IOs Dialog

The "Configure Selected IOs Dialog"allows the user to configure the electrical settings for the currently-selected I/Os in the IO Assignment View (see page 191). Changes made from this dialog will not require that PnR be re-run. See Managing I/Os (see page 501) for more info.



There are a number of fields that may be edited from the dialog - these are shown in the table below. When possible, the choices displayed in the comboboxes will be filtered to show just those choices that are compatible with the rest of the configuration of the selected I/Os.

Table 134: ***Fields shown in the Configure Selected IOs Dialog***

Field	Description
Change IO Standard to	Enabling this option allows the user to change the IO Standard of the selected IOs to a new compatible value.[1)]
Change Drive Strength to	Enabling this option allows the user to change the Drive Strength of the selected IOs to a new value. This property does not apply for Inputs.
Change Slew to	Enabling this option allows the user to change the Slew rate of the selected IOs to a new value. This property does not apply for Inputs.
Change Open Drain to	Enabling this option allows the user to change the Open Drain setting of the selected IOs to a new value. This property does not apply for Inputs.
Change On Die Termination to	Enabling this option allows the user to change the On Die Termination setting of the selected IOs to a new value. This property does not apply for Outputs.
Change Termination Value to	Enabling this option allows the user to change the Termination value of the selected IOs to a new value. This property does not apply for Outputs and only applies if On Die Termination is turned on.
Change Keep Mode to	Enabling this option allows the user to change the Keep Mode setting of the selected IOs to a new value. This property does not apply for Outputs.
Change Hysteresis to	Enabling this option allows the user to change the Hysteresis setting of the selected IOs to a new value. This property does not apply for Outputs.

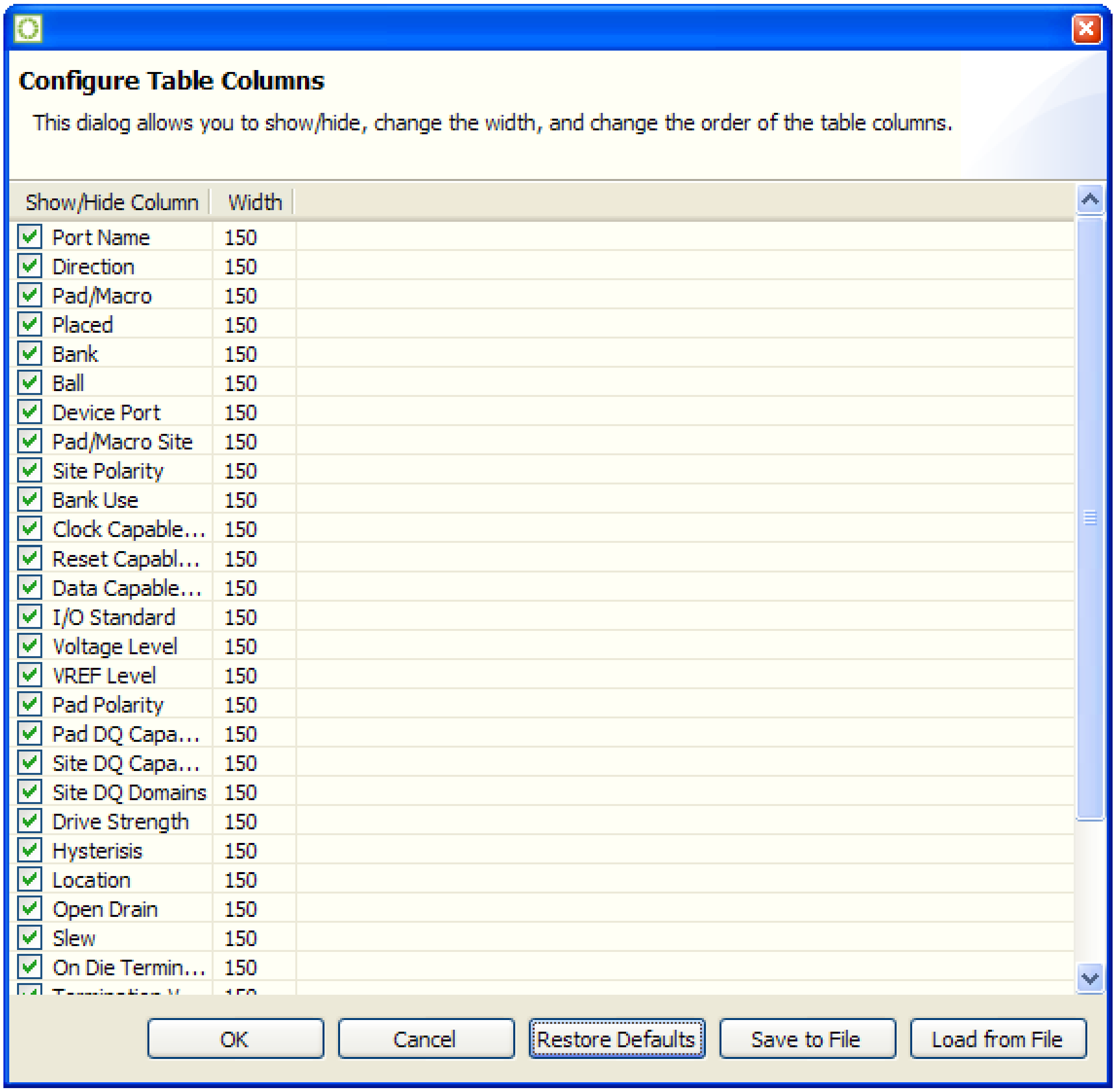
[1)] Some IO Standard changes are not allowed from this dialog, as they would require the re-execution of the **Run Prepare** flow step. If the user wishes to change to an IO Standard that is not listed in the dialog, they must make the change in the source RTL.

Because the dialog is meant to edit multiple I/Os at once, and each I/O will often have different present values for the selected configuration setting, the configuration settings will not initialize to the present configuration values for the selected I/O(s).

See also: IO Assignment View (see page 191) and Managing I/Os (see page 501) for more info.

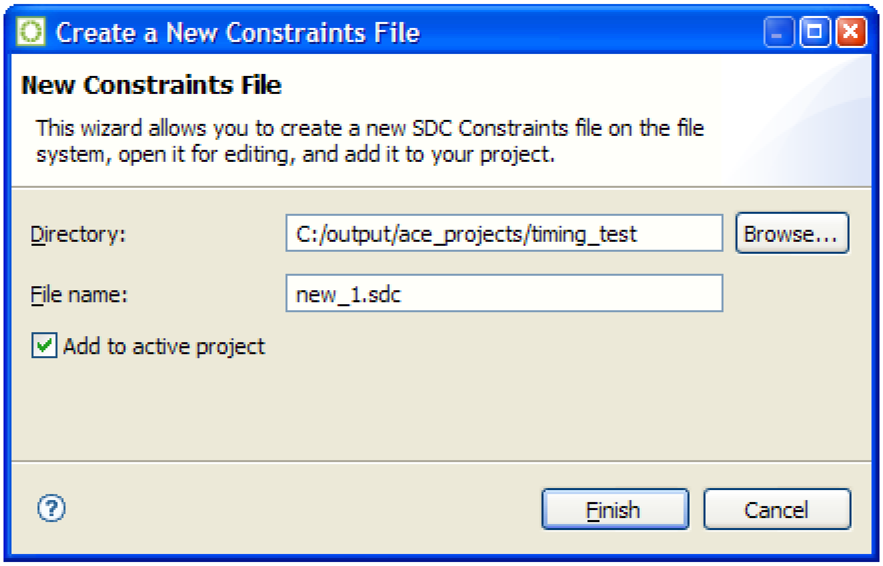
Configure Table Columns Dialog

The "Configure Table Columns Dialog" allows the user to configure the columns shown in the active view. Currently this dialog is only available for the IO Assignment View (see page 191) . From the dialog, the user may configure which columns are visible and the width (in pixels) of each column. The user may also save the current column configuration to a file, or load a previous column configuration from a file.



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…**.



The dialog allows the user to type the file's destination Directory, or select it graphically using the **Browse…** button. The Directory name provided must already exist. (If selected, the **Browse...** button will display a Directory Selection Dialog, which also allows the user to create a new directory and then select it.)

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

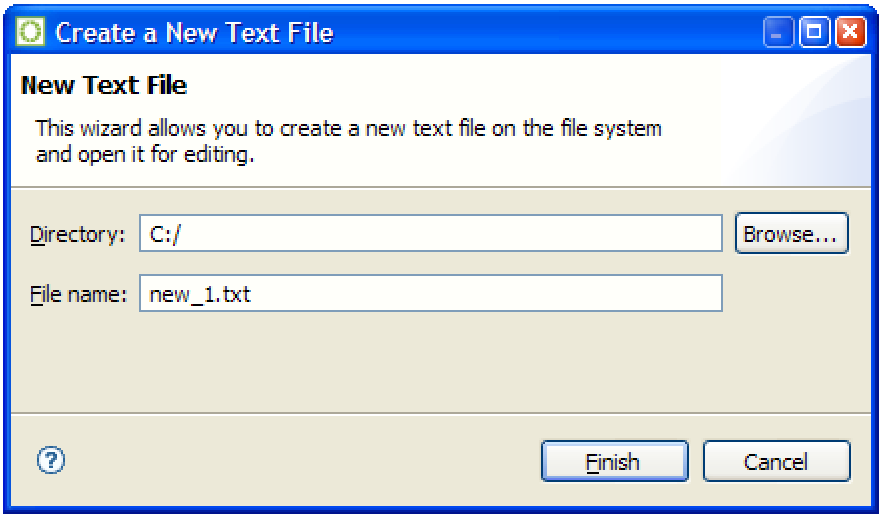
If there is currently an active project in ACE, the **Add to active project** checkbox will be enabled and checked by default. If there is no project active, the checkbox will be disabled and deselected.

Once **Finish** is selected, the text file will be created, and a Text Editor (see page 26) will be opened in ACE for the new text file.

> This dialog may be used to create PDC files as well as SDC files. Simply change the File Name extension to `'.pdc'` instead of `'.sdc'`.

Create a New Text File Dialog

The Create a New Text File Dialog simply allows the user to create a new text file and open it in the ACE text editor in a single action. The dialog is available in all Perspectives (see page 23), and can be selected via **File -> New -> Text File**
.



The dialog allows the user to type the file's destination directory, or select it graphically using the **Browse...** button. The **Directory** name provided must already exist. (If selected, the **Browse...** button will display a Directory Selection Dialog, which also allows the user to create a new directory and then select it.)

The **File name** must be unique - there must not already be a file with that name in the destination directory.

Once **Finish** is selected, the text file will be created, and the ACE Text Editor (see page 26) will be 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.

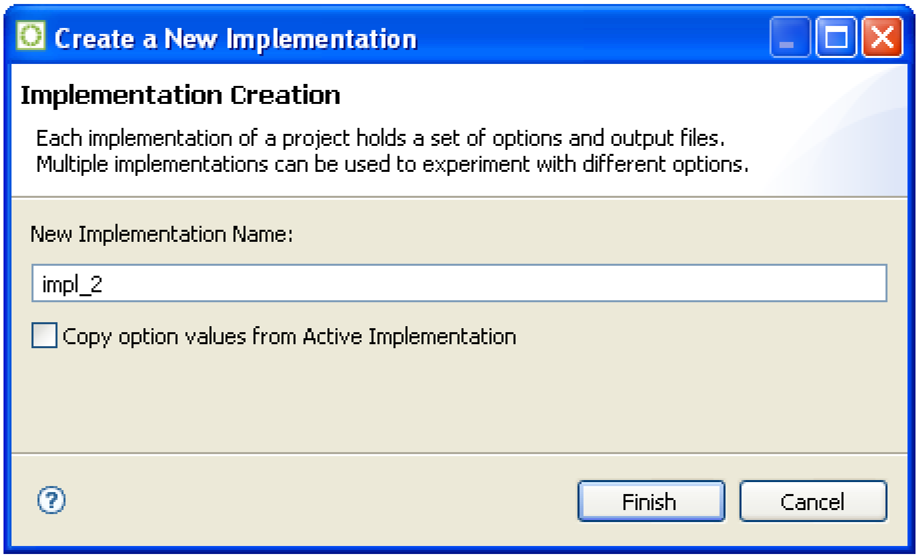


Table 135: *Create Implementation Dialog Fields*

Field	Description	Default
New Implementation Name	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.	impl_
Copy Option Values from Active Implementation	If this field is checked, option values from the current Active Implementation will be copied into the new implementation.	Off

Create Placement Region Dialog

This wizard dialog appears after the user has used drag-and-drop to define a rectangular area in the Floorplanner View (see page 178) while the Placement Region Tool is active. It allows the user to name the new Placement Region, and define its bounds.

See also: Creating a new placement region (see page 436).

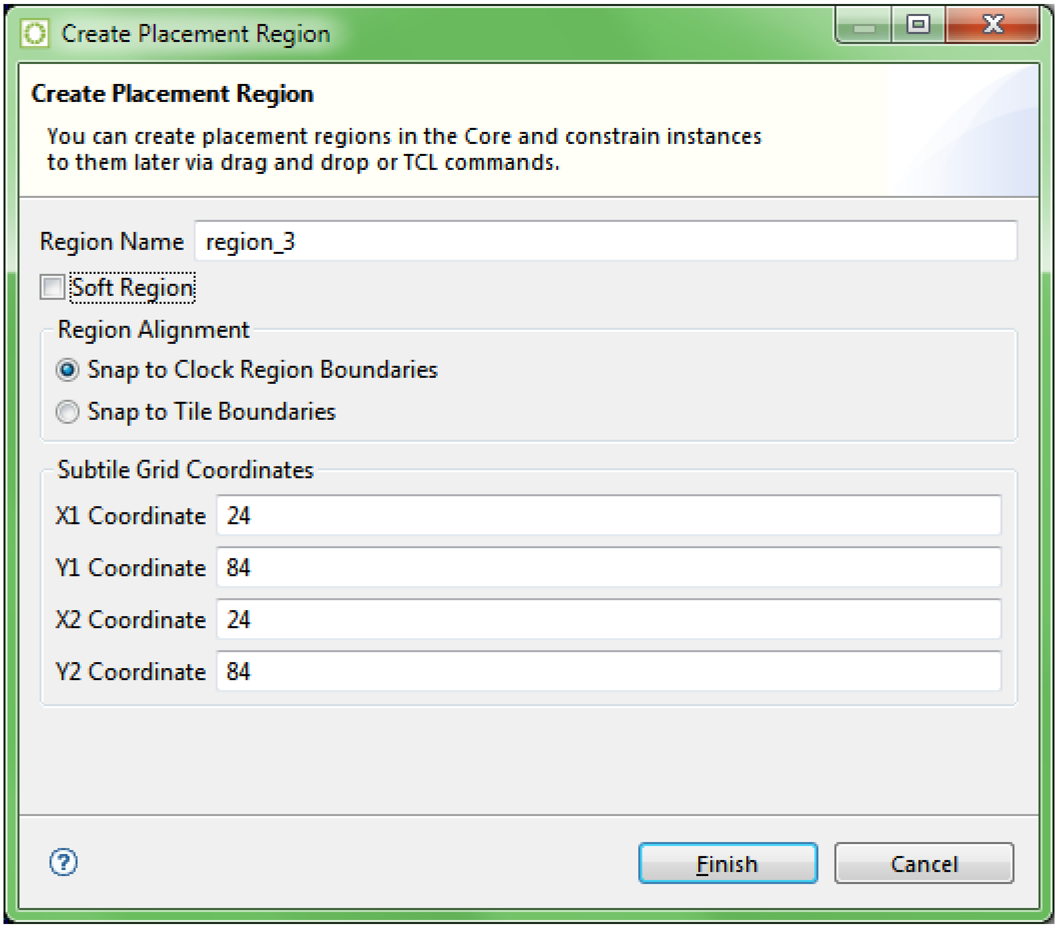


Table 136: *Create Placement Region Dialog*

Option	Description
Region Name	The name for the new Placement Region. ACE will pre-populate this field with a default incrementing value.
Soft Region	If checked, the created Placement Region will be a Soft Placement Region. Soft Placement Regions are rendered as ellipses in the Floorplanner View (see page 178), 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 435) for more details.
Region Alignment	
Snap to Clock Region Boundaries	If selected, ACE will create a Placement Region that encompasses all Clock Regions which contain any of the selected Tiles. Note that in this mode, since Placement Regions can only contain entire Clock Regions (no partial Clock Regions), the Placement Region will almost certainly grow larger than the outline rectangle.

Option	Description
Snap to Tile Boundaries	If selected, ACE will create a Placement Region that encompasses all Tiles selected within the drag-and-drop rectangle. Note that since Placement Regions can only contain entire sites (no partial sites), the Placement Region will potentially grow larger than the outline rectangle.
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.

When using Subtile Grid Coordinates, the 0,0 coordinate will map 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 will vary by device.

Create Project Dialog

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

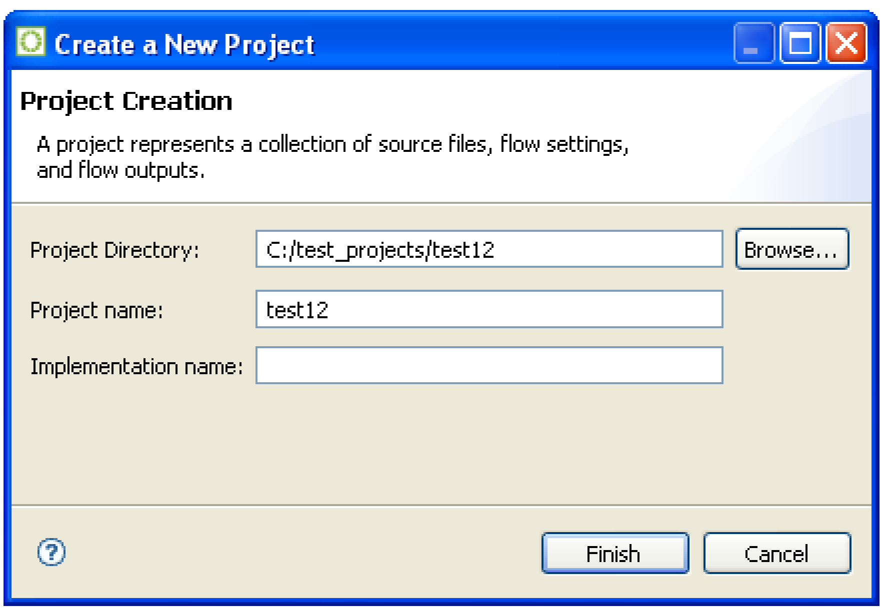
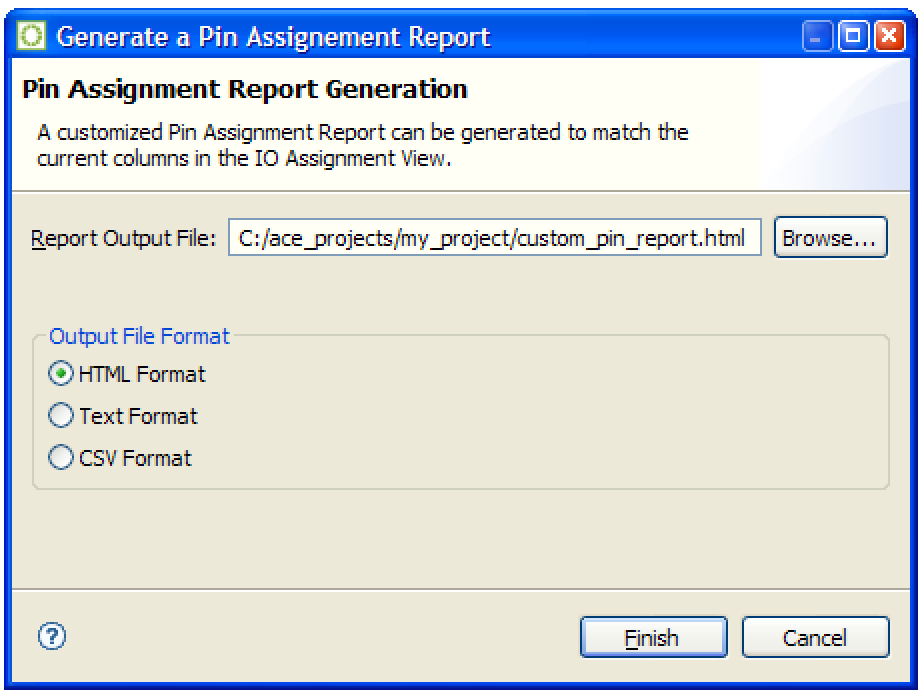


Table 137: *Create Project Dialog Fields*

Field	Description
Project Directory	The location in the file system where the project is created. Users can 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

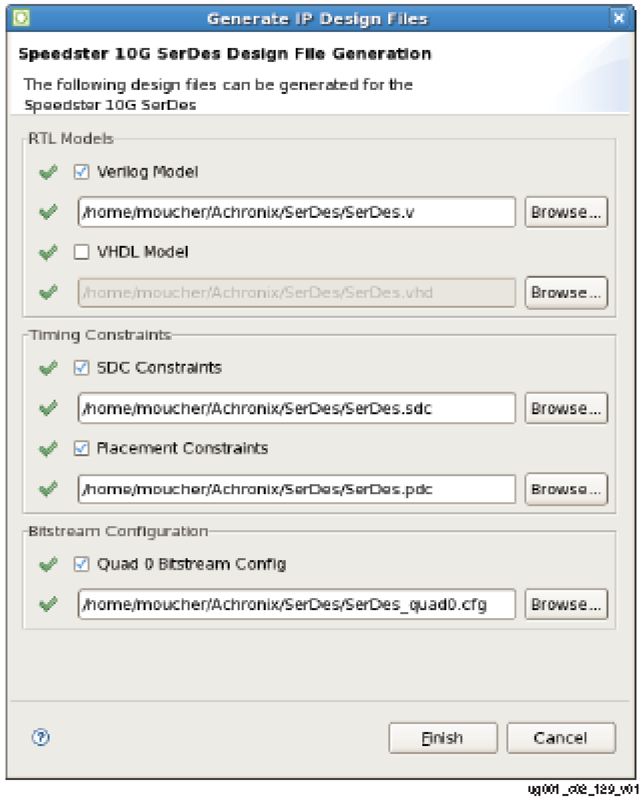
The "Generate a Pin Assignment Report" dialog allows the user to generate a customized Pin Assignment Report (see page 322) with a column organization identical to the current organization of the columns in the "IO Assignment View (see page 191)".



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 25).

See also: Creating an IP Configuration (see page 387).


Generate IP Design Files
Speedster 10G SerDes Design File Generation
The following design files can be generated for the Speedster 10G SerDes
RTL Models
Verilog Model
/home/moucher/Achronix/SerDes/SerDes.v
Browse...
VHDL Model
/home/moucher/Achronix/SerDes/SerDes.vhd
Browse...
Timing Constraints
SDC Constraints
/home/moucher/Achronix/SerDes/SerDes.sdc
Browse...
Placement Constraints
/home/moucher/Achronix/SerDes/SerDes.pdc
Browse...
Bitstream Configuration
Quad 0 Bitstream Config
/home/moucher/Achronix/SerDes/SerDes_quad0.cfg
Browse...
Finish
Cancel

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 138: *Generate IP Design Files Dialog Fields*

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

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's design

.

Load Project Dialog

The Load Project dialog is used to browse 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.

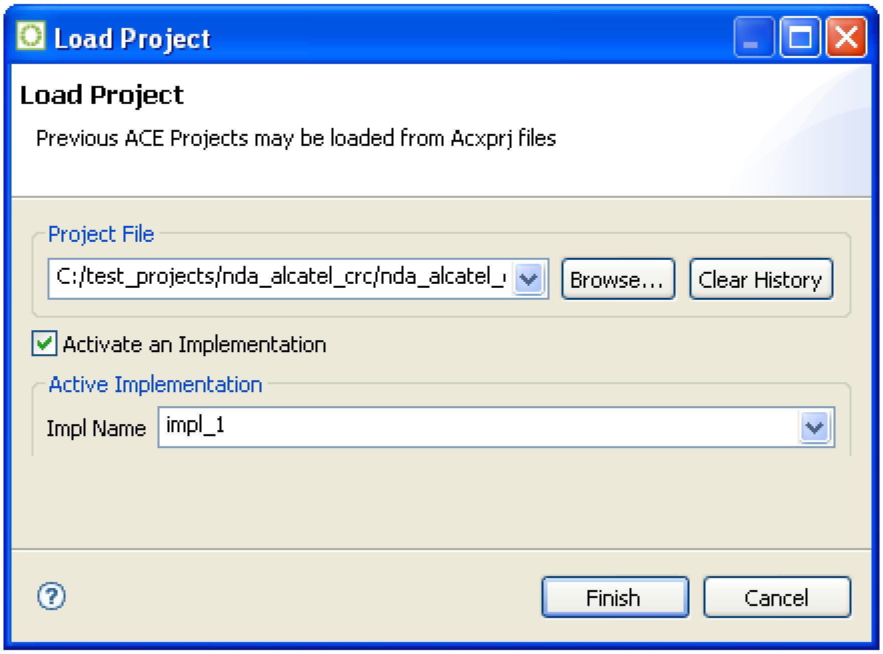
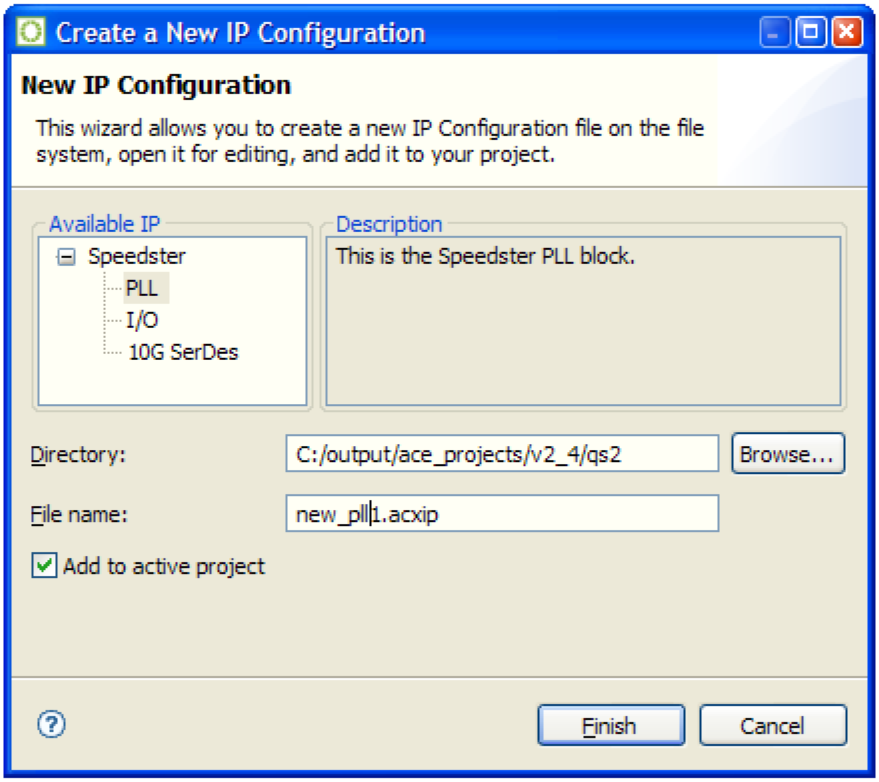


Table 139: *Load Project Dialog Fields*

Field	Description
Project File	The file path to the ACE Project File (.acxprj) to load. Users can 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 on the Clear History button.
Activate an Implementation	The user can choose to activate an implementation upon loading the project. If another project is already loaded and this field can be un-checked to preserve the current active implementation in the ACE session.
Implementation Name	The name of the implementation to activate after loading the project. A drop-down list allows the user to select from any implementation defined within the specified project file.

New IP Configuration Dialog

The New IP Configuration dialog helps users 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 25). See also: Creating a New IP Configuration (see page 387).



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 by the ACE end user.

Table 140: *New IP Configuration Dialog Fields*

Field	Description	Default
Available IP	Lists the available IP blocks by FPGA family.	–
Description	Provides a description of the IP block.	–
Directory	The location in the file system where the configuration file will be created. Users can either type the new location or browse to select a file system location for the new configuration.	*(current active project directory)*
File Name	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. ⚠ Names that collide with Achronix's reserved module names are prohibited.	`new_ . acxip`
Add to active project	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.	Enabled

Restore Implementation Dialog

The Restore Implementation dialog is used to retore the database state of the active implementation from an Acxdb Archive File. After indicating the file path to the Acxdb Archive File to restore the implementation from, click **Finish** to restore the active implementation.

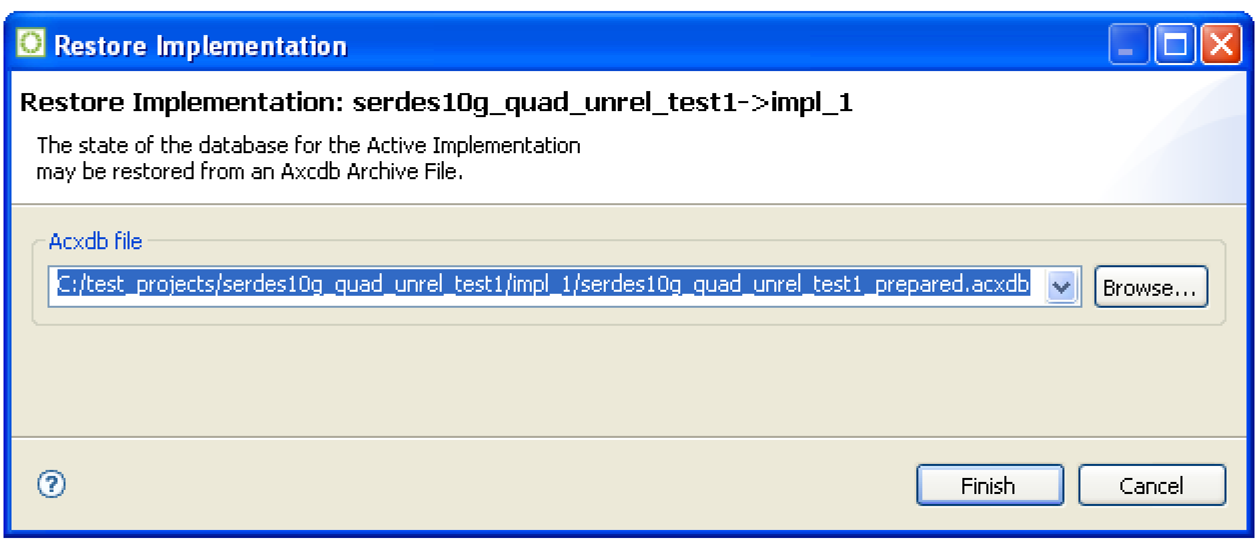
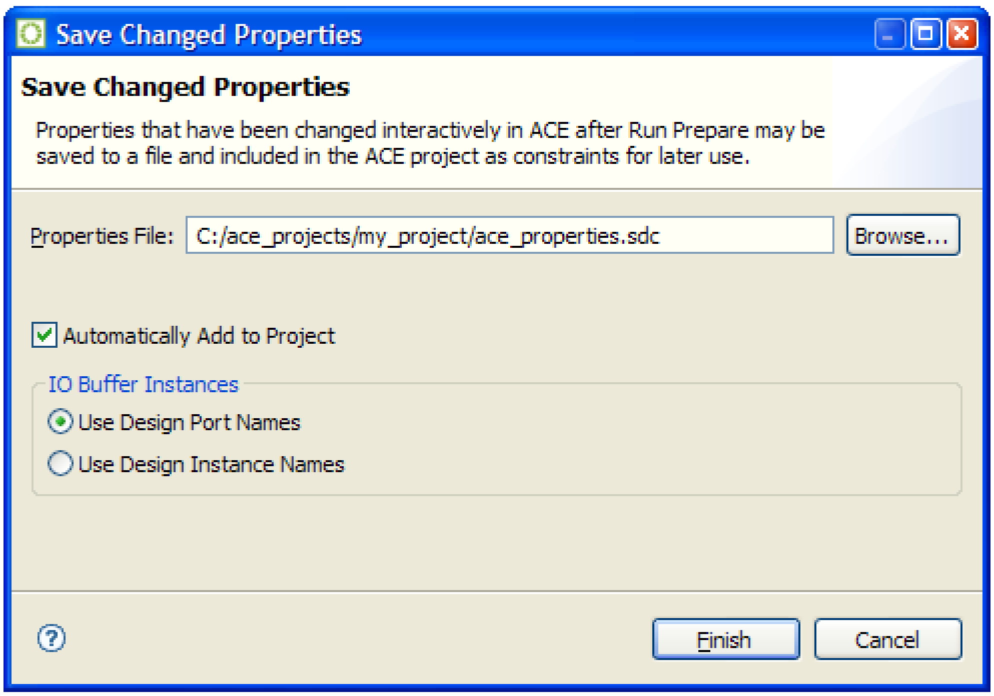


Table 141: ***Restore Implementation Dialog Fields***

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

Save Changed Properties Dialog

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



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

See also: IO Assignment View (see page 191) and Managing I/Os (see page 501) .

Save Implementation Dialog

The Save Implementation dialog is used to save the database state of the active implementation to an Acxdb Archive File. After indicating the file path to the Acxdb Archive File to save the implementation to and whether to include the log file, click **Finish** to save the active implementation.

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.

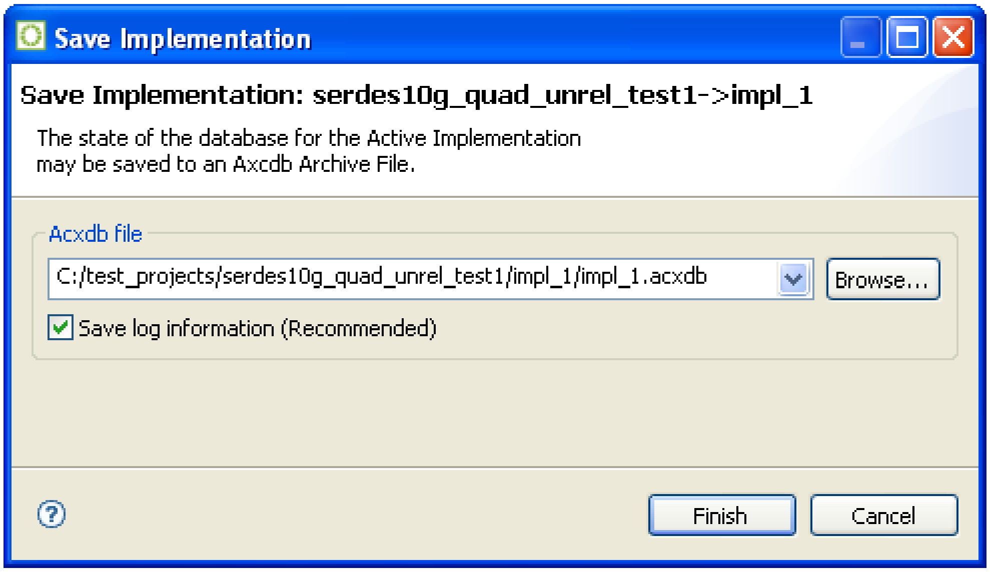


Table 142: *Save Implementation Dialog Fields*

Field	Description	Default
Acxdb File	The file path to the Acxdb Archive File to save the implementation to.	.acxdb
Save Log Information	If this field is checked, the log file for the current Active Implementation will be included in the Acxdb file.	On

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.

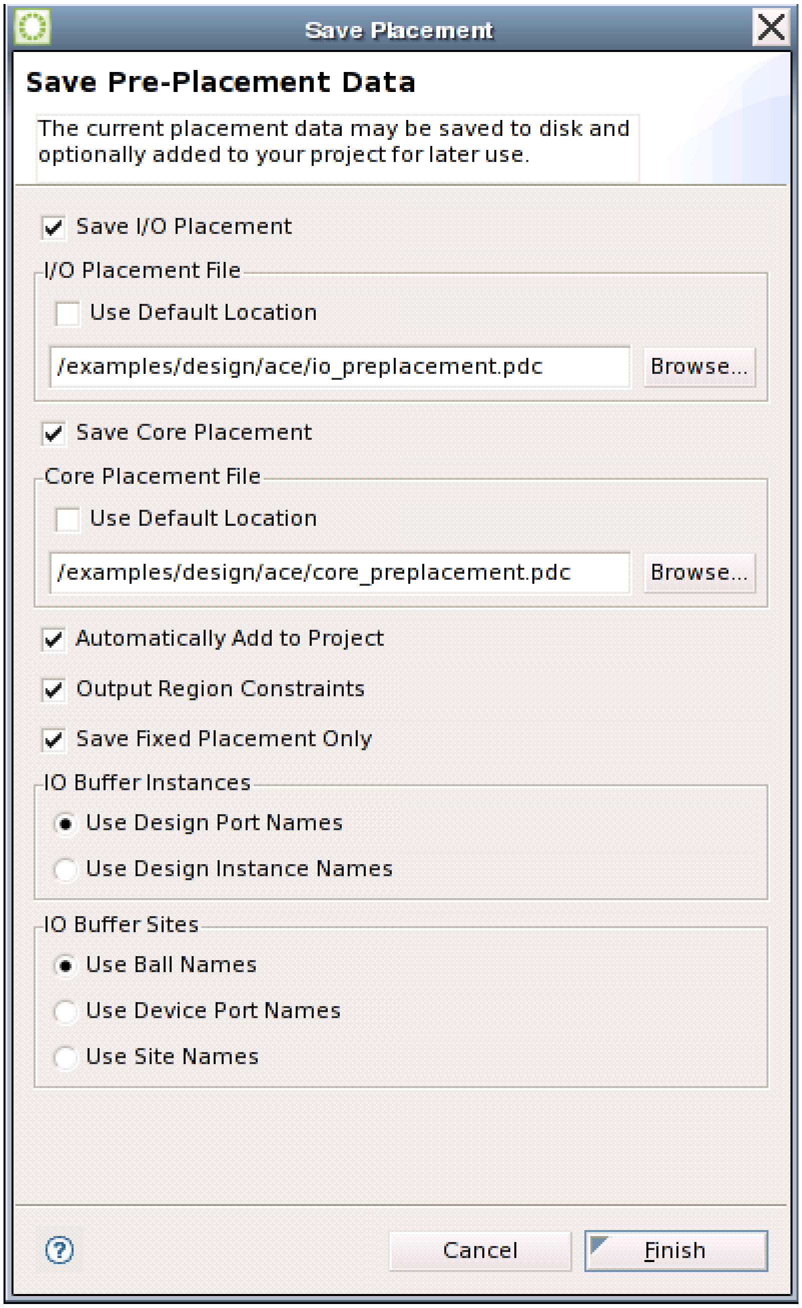


Figure 30: ***Screenshot of Save Placement Dialog***

Table 143: ***Save Placement Dialog Fields***

Field	Default	Description
Save I/O Placement	Enabled	This option indicates whether placement of instances in the I/O ring should be saved.
I/O Placement File		

Field	Default	Description
Use Default Location	Enabled	Selects whether the default I/O placement file path is used, or the one manually specified below the option.
Save Core Placement	Enabled	This option indicates whether placement of instances in the core fabric should be saved.
Core Placement File		
Use Default Location	Enabled	Selects whether the default core placement file path is used, or the one specified below the option.
Automatically Add to Project	Enabled	This option 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 435) are exported to the PDC file.
Save Fixed Placement Only[1]	Enabled	This option 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) will be ignored.
IO Buffer Instances	Use Design Port Names	Selects whether the placement file uses I/O buffer names based on port names of the HDL source, or instance names generated during synthesis.
IO Buffer Sites		
Use Ball Names	Enabled	This option controls the output format of the site names for I/O buffers. Selecting this option outputs the package ball name instead of the site name on the device.
Use Device Port Names	Disabled	This option controls the output format of the site names for I/O buffers. Selecting this option outputs the top-level device port name instead of the site name on the device.
Use Site Names	Disabled	This option controls the output format of the site names for I/O buffers. Selecting this option outputs site name on the device.

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

Save Placement Regions Dialog

This wizard dialog appears after the user has selected the "Save Placement Regions" action in the Placement Regions view. It allows the user to save placement region definitions, including the instance constraints for those placement regions.

See also: Saving Placement Region Constraints (see page 441) .

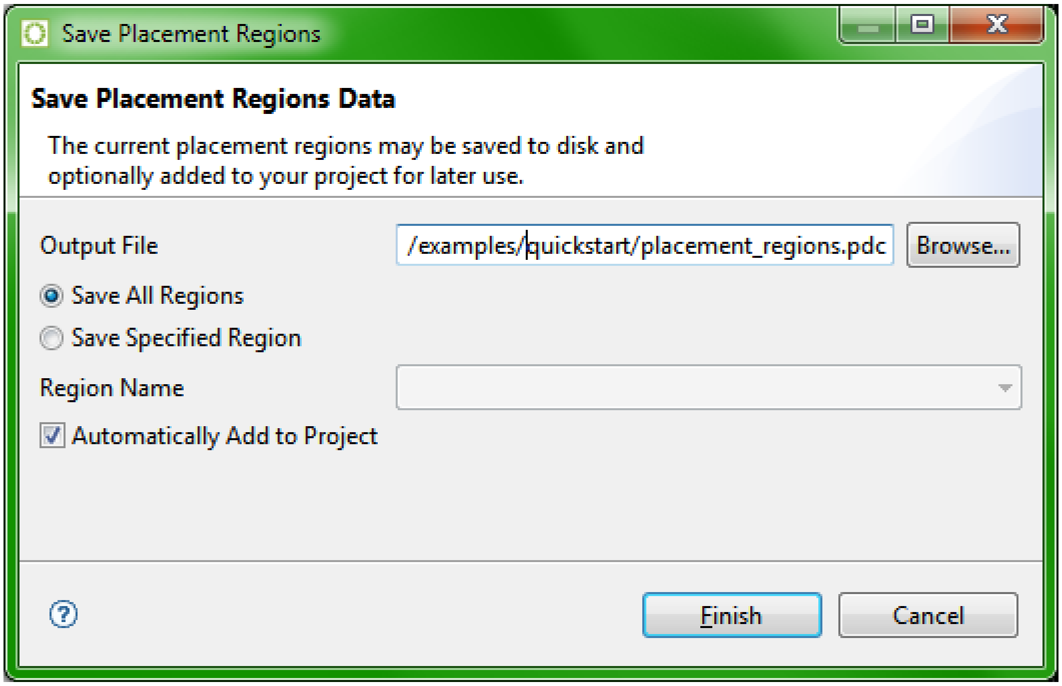


Table 144: *Save Placement Regions Dialog*

Option	Description
Output File	The full path to the pdc file which will contain the region definitions and constraints.
Save all Regions	This option will save the data for all the regions listed in the Placement Regions view.
Save Specified Region	This option will save the data for a single region (specified below).
Region Name	The name of the Placement Region to be saved. This 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 is shown when the user selects the ( ) **Write Script for Schematic View** button in the Critical Paths View (see page 174).

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 **Save** to write the script to disk, or **Cancel** to close the dialog without saving.

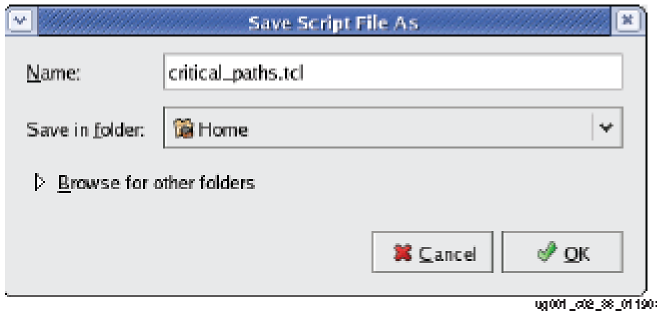


Figure 31: Unix Screenshot of the Save Script File As dialog

Search Filter Builder Dialog

This wizard dialog allows the user to build simple or compound search filters to be used in the Search View (see page 245). (It 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 329) for a table of the available filter properties with descriptions.

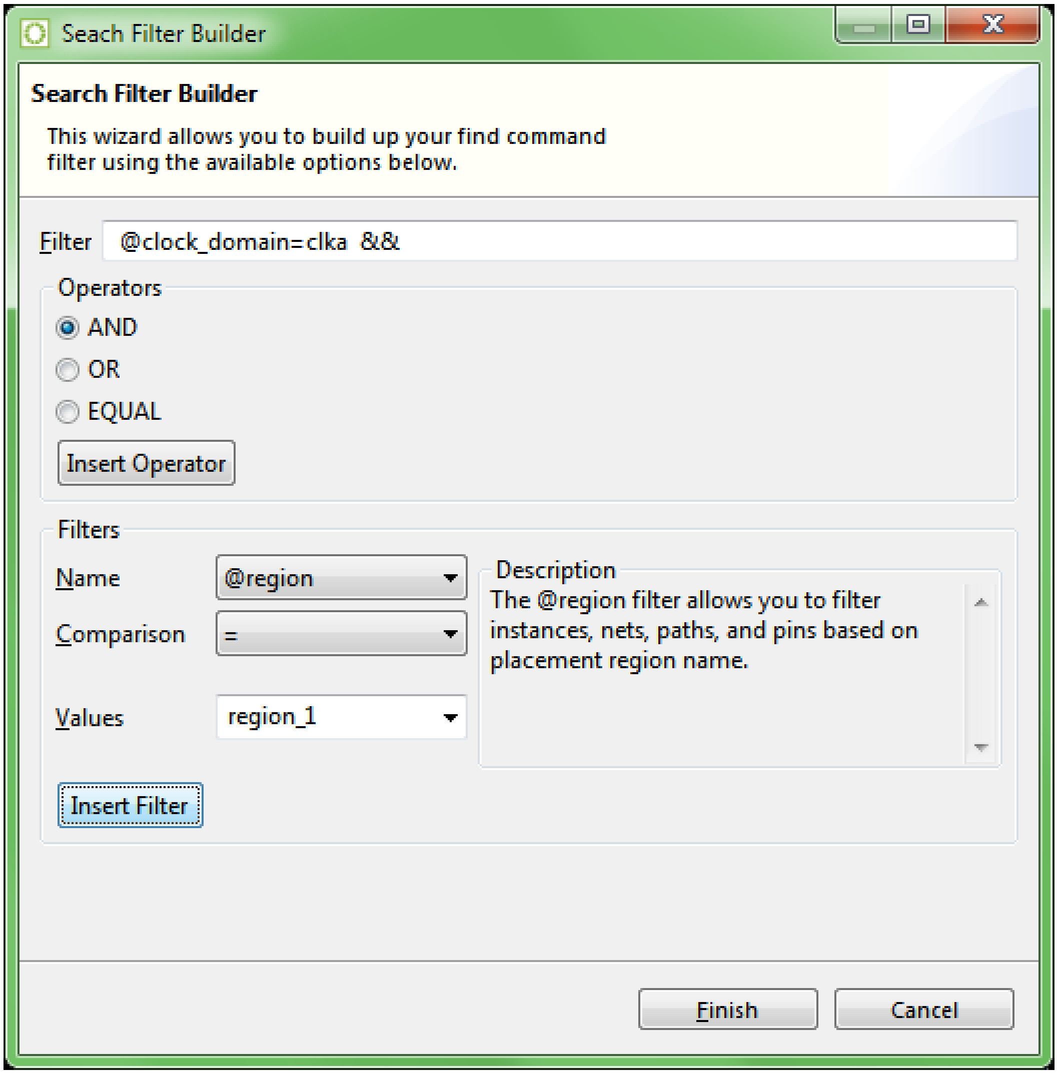


Figure 32: ***Screenshot of the Search Filter Builder Dialog***

Table 145: ***Search Filter Builder Dialog Options***

Option	Description
Filter	The filter string itself. The user may type directly into this field, or may populate this field by the use of the **Insert Operator** and **Insert Filter** buttons.
Finish	Press this button to close the dialog, copying the current value of the **Filter** text field into the **Filters** field of the Search View.
Cancel	Press this button to close the dialog without changing the current value of the **Filters** field of the Search View.
Operators	
AND	Select this radio button when you want to join two filters into a compound filter where both sub-filters are true.

Option	Description
OR	Select this radio button when you want to join two filters into a compound filter where either sub-filter is true.
EQUAL	Select this radio button when you want to join two filters into a compound filter where the Boolean value of both sub-filters is identical.
Insert Operator	Press this button to insert the selected Boolean operator into the **Filter** field at the current text cursor position within that field.
Filters	
Name	This is a combo box showing all the 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 329).
Description	Provides a textual description of the current filter parameter selected in the **Name** field. This may also provide hints or details on how the **Comparison** or **Values** fields may be populated.
Comparison	Allows the user to select from the set of possible comparisons relevant to the selected filter parameter **Name**. The contents of this combo drop-down change according to the current **Name** selection.
Values	Allows the user to type in the desired value(s) to compare against. For some filter parameter **Name**s, the combo drop-down may contain possible values. The contents of this combo drop-down change according to the current **Name** selection.

Toolbars

There are three kinds of toolbars in the Workbench (see page 23): 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 163) or editor (see page 25). 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 163). Actions in a view's 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 will also produce a toolbar in the trim at the outer edge of the workbench window (a Trim Stack). This bar will contain an icon for each of the views in the stack and/or a single icon for each stack of editors. Clicking on one of these icons will result 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 is used to set user preferences. The Preferences 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

This page configures the Preferences (see page 287) for the DCC (Demo Command and Control) connection, as used for the HW Demo View (see page 189).

See also: Configuring the DCC Connection (see page 410), Running the HW Demo (see page 441).

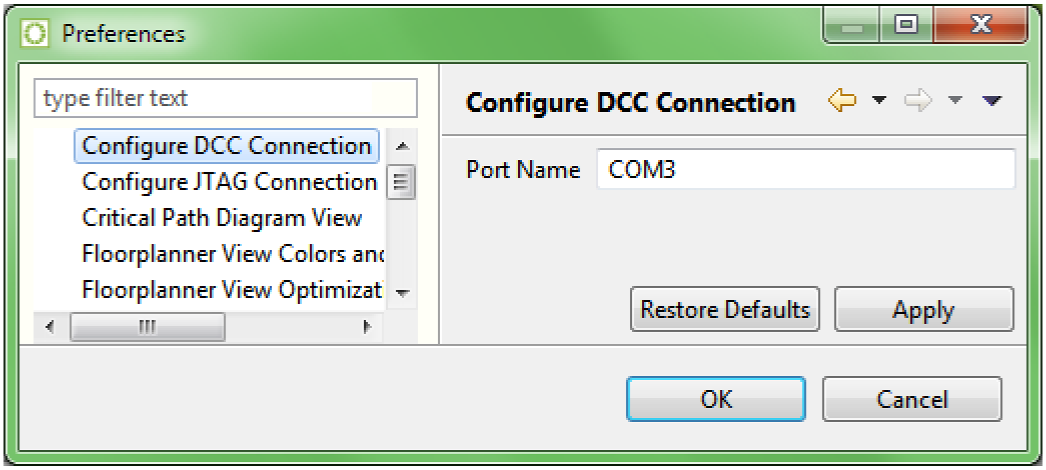


Table 146: ***Configure DCC Connection Preference Page Options***

Option	Description
Port Name	Enter the serial port name which will be used for the DCC connection. For further information about determining which serial port should be used, please see Configuring the DCC Connection (see page 410).

Configure JTAG Connection Preference Page

This page configures which Bitporter or FTDI FT2232H device will be used to talk via JTAG to the desired Device Under Test (the test board). It also specifies where the Achronix device will be found in the JTAG scan chain, which may potentially contain multiple Achronix and non-Achronix devices.

Warning!

Bitporter JTAG pods may be damaged if connected improperly. Before attempting to use a Bitporter JTAG pod, please consult the *Bitstream Programming and Debug User Guide (UG004)*.

Multiple views will use these configuration preferences, including the Download View (see page 176), the Snapshot Debugger View (see page 253), the JTAG Browser View (see page 199), and the HW Demo View (see page 189). Specialized functionality for some IP Configuration Editors may also use these JTAG preferences.

The section Configuring the JTAG Connection (see page 411) explains the proper use of all fields of this page in detail.

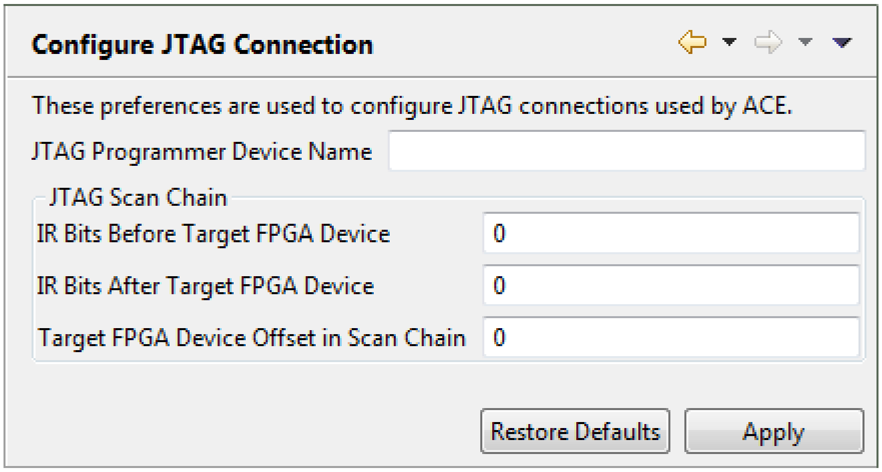


Figure 33: ***Configure JTAG Connection Preference Page***

Table 147: ***Configure JTAG Interface Preference Page Options***

Option	Description
JTAG Programmer Device Connection	
JTAG Programmer Device Name	The name of the JTAG programmer device which should be used for all of ACE's JTAG interactions with the chosen FPGA. If this is not specified, auto-detection of JTAG programming devices will be attempted. ⚠ 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 the user is required to specify which pod/device to use. ✓ 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.(†)
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.(†)
Target FPGA Device Offset in Scan Chain	Sets the device count (in decimal) between the board JTAG TDI pin and target FPGA device.(†)

Option	Description

Table Note

† 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 needs to 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.

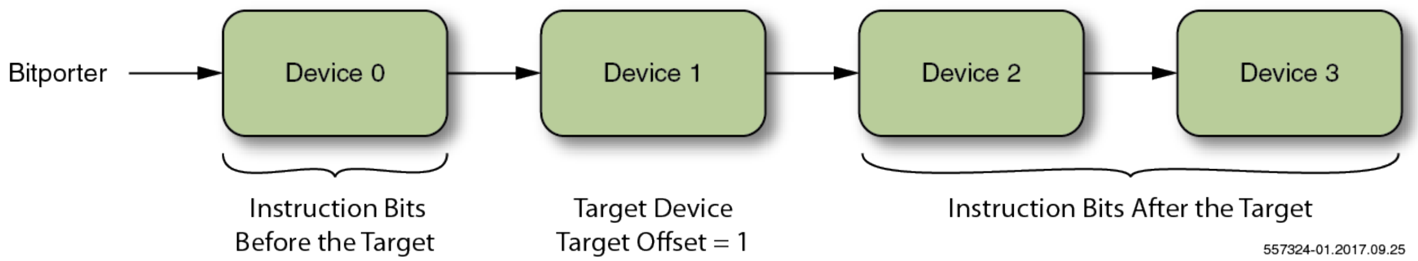


Figure 34: ***Example High-Level Diagram of a Multi-device JTAG Scan Chain***

When multiple FPGA devices are attached to the same JTAG scan chain, the user must specify which FPGA device is the target. The FPGA device closest to the Bitporter (more accurately, closest to the board's JTAG TDI pin) has target offset 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 411).

Critical Path Diagram View Preference Page

This page configures the display Preferences (see page 287) of the Critical Path Diagram View (see page 170).

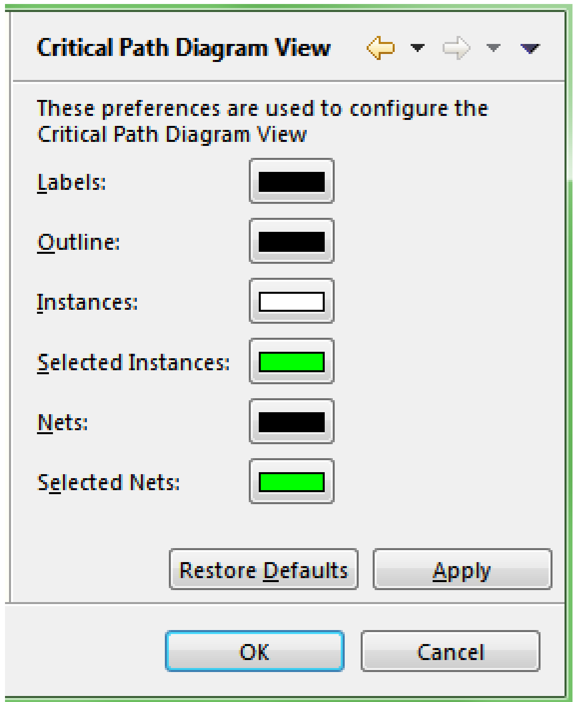


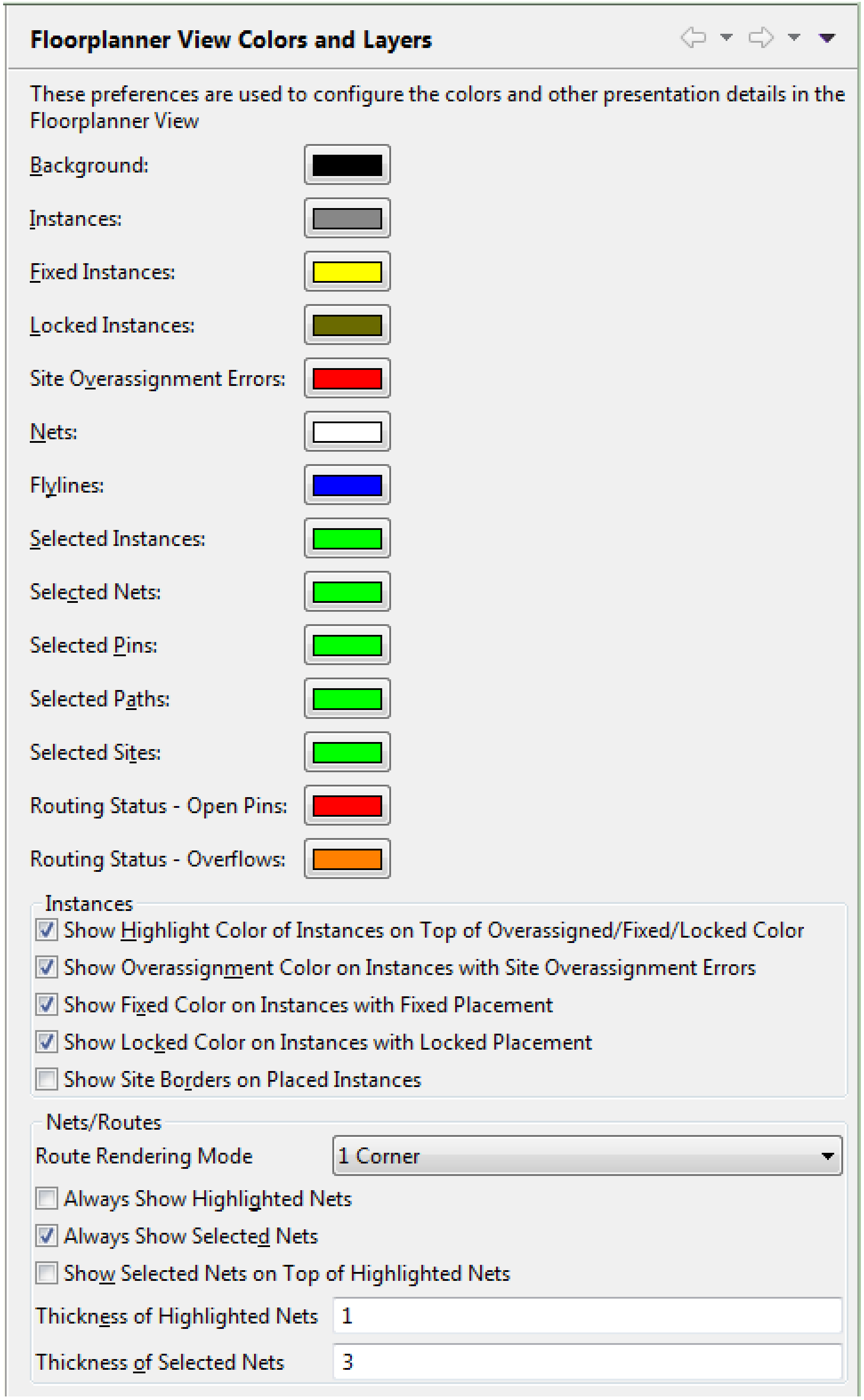
Table 148: *Critical Path Diagram View Preference Page Options*

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	When this button is pressed, all preferences on this page are returned to their ACE default values.
Apply	When this button is pressed, any configuration changes on this page are immediately applied to the current diagram in the Critical Path Diagram View (see page 170). These config values are also saved and will be used in all future ACE sessions. The Preferences dialog stays open to allow other preference configuration changes if desired.
OK	When this button is pressed, any preference configuration changes (including on other preference pages) are immediately applied. These config values are also saved and will be used in all future ACE sessions.

Option	Description
Cancel	When this button is pressed, any preference configuration changes made since the dialog was opened (or since the last time an Apply button was pressed in the dialog, whichever was most recent) are discarded.

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 178). Additionally, options are provided allowing the user a degree of control over the display priorities of the possible Instance States (see page 328) and Net/Route highlighting vs. selection. (For more about Highlighting, see Highlighting Objects in the Floorplanner View (see page 396). For more about Selection, see the Selection View (see page 249).)


Floorplanner View Colors and Layers
These preferences are used to configure the colors and other presentation details in the Floorplanner View
Background:
Instances:
Fixed Instances:
Locked Instances:
Site Overassignment Errors:
Nets:
Flylines:
Selected Instances:
Selected Nets:
Selected Pins:
Selected Paths:
Selected Sites:
Routing Status - Open Pins:
Routing Status - Overflows:
Instances
Show Highlight Color of Instances on Top of Overassigned/Fixed/Locked Color
Show Overassignment Color on Instances with Site Overassignment Errors
Show Fixed Color on Instances with Fixed Placement
Show Locked Color on Instances with Locked Placement
Show Site Borders on Placed Instances
Nets/Routes
Route Rendering Mode
1 Corner
Always Show Highlighted Nets
Always Show Selected Nets
Show Selected Nets on Top of Highlighted Nets
Thickness of Highlighted Nets
1
Thickness of Selected Nets
3

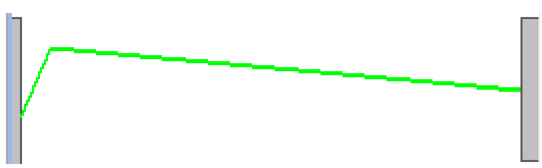
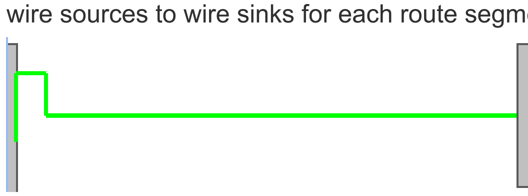
Color Preference	Description
Background	Used to render the background of the device.
Instances	Represents instances with default (Soft) placement.
Fixed Instances	Represents instances with Fixed placement.
Locked Instances	Represents instances with Locked placement.
Site Overassignment Errors	Represents 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's 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 249)).
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 will only be visible when enabled in the Layers section of the Floorplanner view's Palette.
Routing Status - Overflows	Represents Overflow pins, a rare routing error state. Overflow diamonds on pins will only be visible when enabled in the Layers section of the Floorplanner view's Palette.

The meanings of the various Instance States (see page 328) are defined elsewhere. While Instances can have multiple states at once, they're only able to show a single state at a time, thus the user is provided some ability to alter the display priority of the various Instance states.

Instance Preference	Description
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.

Instance Preference	Description
Show Overassignment Color on Instances with Site Overassignment Errors	Allows the user to toggle 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 will be shown in the same color, grey by default.
Show Locked Color on Instances with Locked Placement	If disabled, locked and non-locked instances will be shown with the same color.
Show Site Borders on Placed Instances	If enabled, all placed instances will be rendered with the Site border color as an outline around the instance. If disabled, placed instances will be rendered without a site border. When this is enabled, when the view is extremely zoomed out, the site border render color may actually hide the placement state color. Thus, this is disabled by default.

Similarly, users are able to tweak the display of the Floorplanner's Nets/Routes layers. (Both Clock and non-Clock nets will be affected identically by these settings.)

Nets / Routes Preference	Description
Route Rendering Mode	Allows the user to alter 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's Y coordinate, then a horizontal line to the wire sink; no diagonal lines are used, which speeds rendering in complex designs. **Performance Tip:** The **1 Corner** route drawing mode is significantly faster (up to 5x) 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.

Nets / Routes Preference	Description
Always Show Highlighted Nets	If enabled, Highlighted nets will always be displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Highlighted nets will only be displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Always Show Selected Nets	If enabled, Selected nets will always be displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Selected nets will only be displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Show Selected Nets on Top of Highlighted Nets	Allows the user to change 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 will be rendered this many pixels wide.
Thickness of Selected Nets	The selection color of a net will be rendered this many pixels wide.

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 will be rendered as a "halo" effect around the top state's color.

Having the "Always Show..." preferences enabled may be most useful in designs with massive numbers of nets /routes. The user can disable both the Clock and Non-clock route layers, then Select or Highlight the interesting routes, and only the Selected and/or Highlighted routes will be displayed, without the distractions of the less-interesting routes cluttering up the view.

Floorplanner View Optimizations Preference Page

The Floorplanner View Optimizations Preference Page configures rendering optimizations for the Floorplanner View (see page 178).

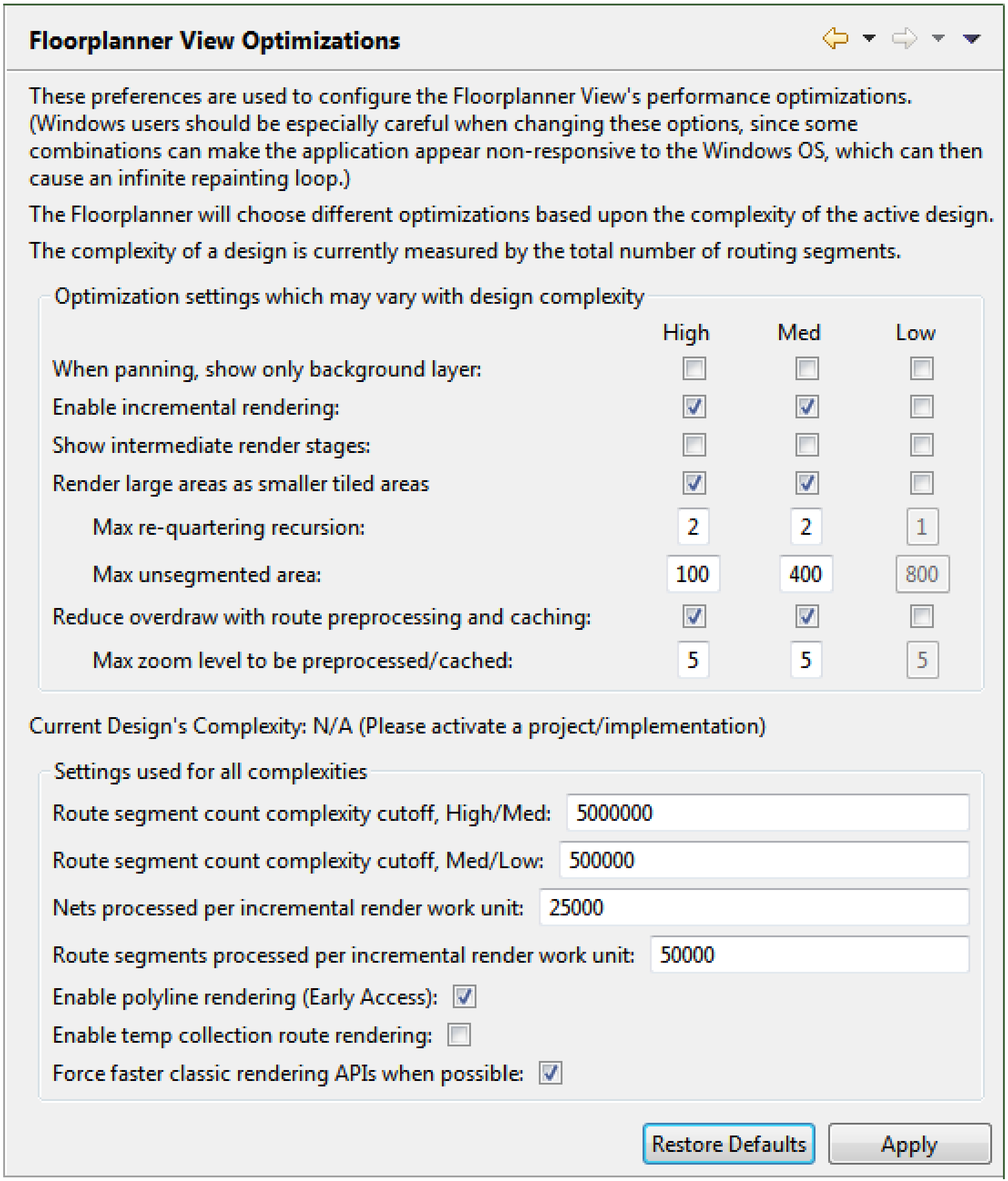


Figure 35: Screenshot of Floorplanner View Optimizations, showing default values for Windows7

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's optimization settings in the 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 may be configured by the user.

By default, all optimization features are disabled for the simplest designs. As design complexity increases, more optimizations will be enabled by default. Note that some optimizations have overhead, and will 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's Floorplanner complexity is always reported in this preference page as **Current Design's Complexity:** (near the middle of the page). This allows users to know which column of optimization settings will impact the rendering of the current design in the Floorplanner. Be aware that the route segment count used to compute the design's complexity only includes route segments of routed nets. The same design will often report a Low complexity before it is routed, and a High complexity after routing is complete.

Table 149: *Optimization settings which may vary with design complexity*

Option	Technical Description	Usability Notes
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 will be slightly (5% to 10%) slower overall, but the Floorplanner Perspective becomes significantly more responsive to mouse and keyboard actions. Obsolete renders may be then 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 may 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 will actually be slightly slower overall.
Render large areas as smaller tiled areas:	The total render area, if greater than the **Max unsegmented area**, will be broken into quadrants which are rendered individually. Rendered areas will be 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 will increase total render times, but it may 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 may increase total render times, but can provide more frequent visual feedback.
Max unsegmented area:	Areas larger than this will be 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.

Option	Technical Description	Usability Notes
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 will slightly increase the memory footprint & 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 may more than double at each increasing zoom level.)

Table 150: ***Settings used for all complexities***

Option	Description
Route segment count complexity cutoff, High /Med:	Designs with a route segment count greater than this number will 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 will 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. Increasing this number may very slightly improve rendering performance, but will decrease the frequency of visual feedback. Only relevant when **Enable incremental rendering** is on.
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. Increasing this number may very slightly improve rendering performance, but will decrease the frequency of visual feedback. Only relevant when both **Enable incremental rendering** and **Reduce overdraw with route preprocessing and caching** are on.
	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 will be most noticeable in the largest designs.

Option	Description
Enable polyline rendering (Early Access)	**Early Access Functionality** This is new, Early Access functionality. While Achronix found no negative stability or performance impacts when testing this optimization, we know our users have a wider variety of hardware and software configurations than we do in our test labs. If you see any new performance or stability issues in the GUI, please contact Achronix Technical Support. Achronix will note the specifics of your configuration (so we can reproduce and fix the problem), and we may then suggest that you disable this new polyline rendering functionality to see if your performance or stability improves.
Enable temp collection route rendering:	May slow rendering when using route overdraw reduction cache, but other route rendering may speed up slightly. CAUTION: Requires significantly more ACE GUI memory when enabled!
Force faster classic rendering APIs when possible:	The classic rendering APIs are significantly faster in all tested cases, but may produce slightly cruder visual output due to a lack of anti-aliasing. (The modern/advanced render APIs will still automatically be used when absolutely required.)

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 may 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.

IP Diagram Preference Page

There are a number of preferences for the IP Diagram View (see page 196) relating to colors and fonts.

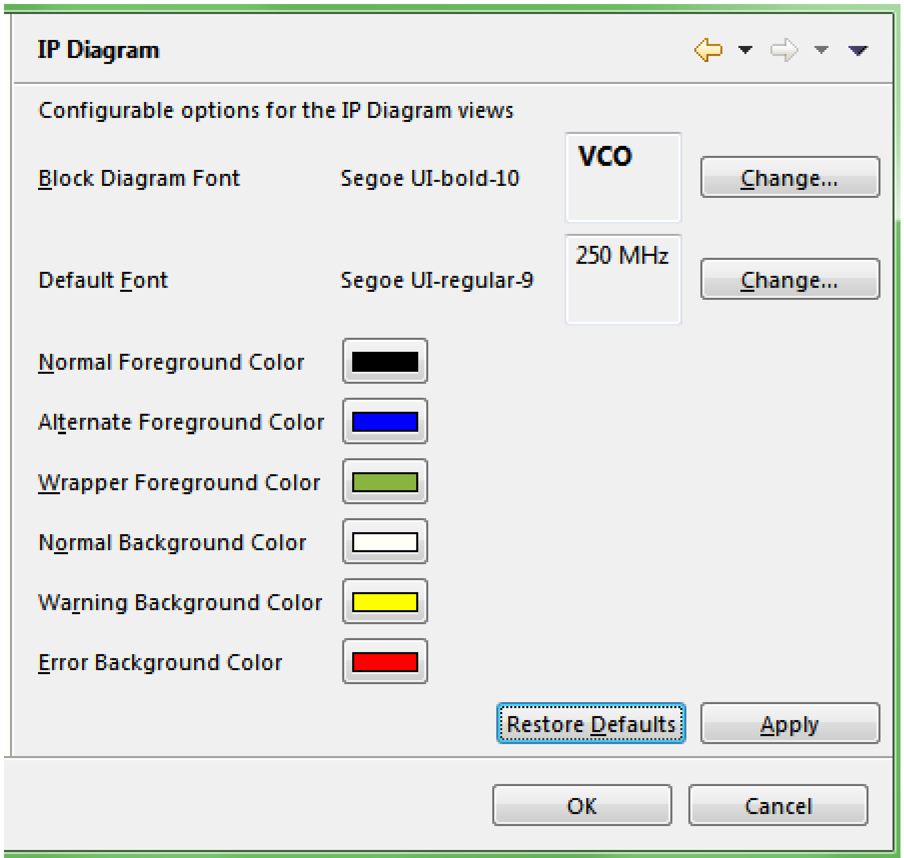


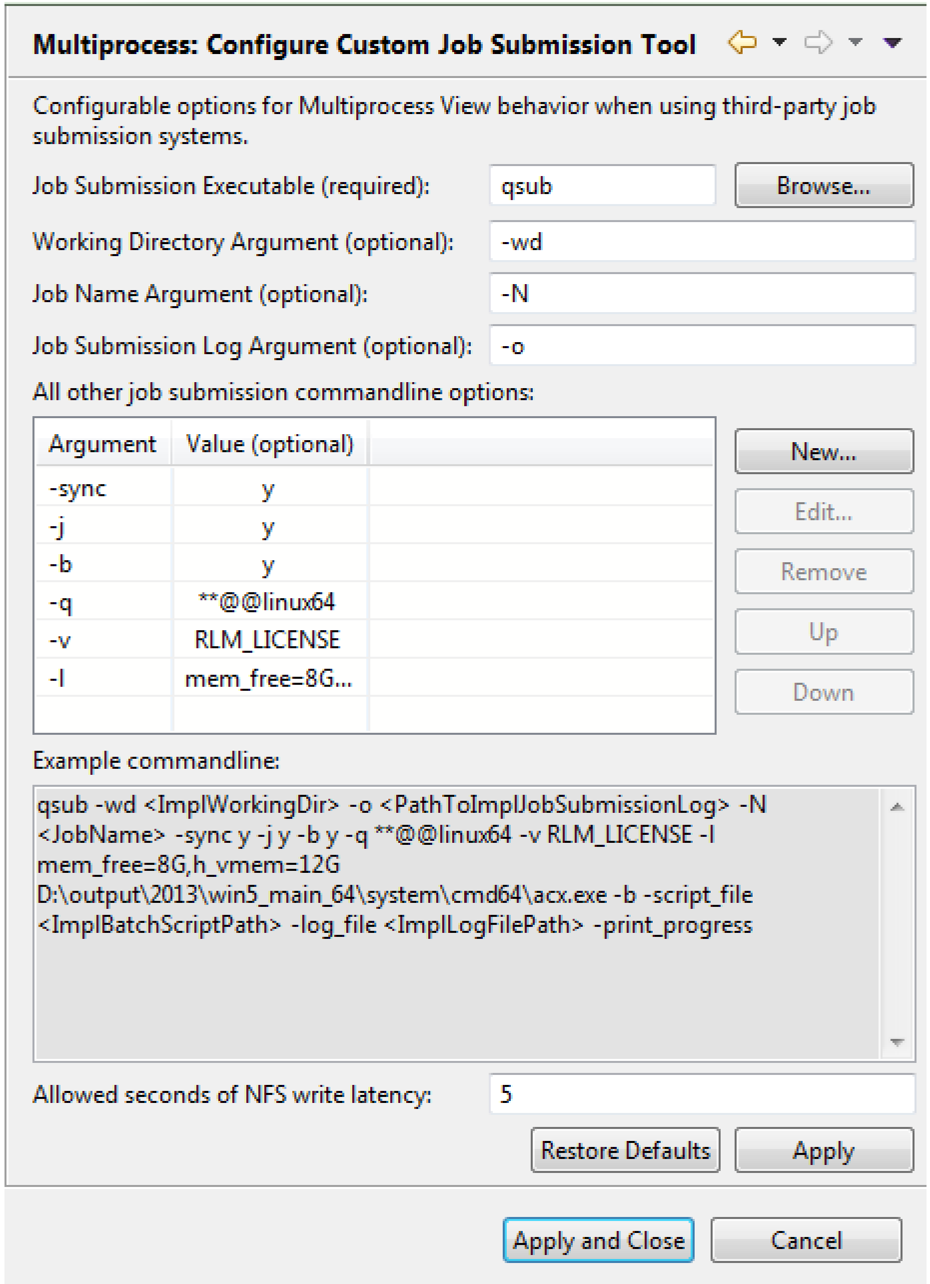
Table 151: *IP Diagram Preferences Options*

Option	Description
Block Diagram Font	This will be the font used to title logic blocks in the diagram.
Default Font	This font will be used for all diagram text except the logic block titles.
Normal Foreground Color	This color will be used for logic blocks, signals, and text.
Alternate Foreground Color	This color will used to highlight portions of the diagram.
Wrapper Foreground Color	This color will be used to represent the IP's RTL wrapper itself. Everything enclosed by this color is within the wrapper.
Normal Background Color	This will be the default background color for the entire diagram.
Warning Background Color	Text representing IP Options with warnings will have their backgrounds painted this color.

Option	Description
Error Background Color	Text representing IP Options with errors will have their backgrounds painted this color.

Multiprocess: Configure Custom Job Submission Tool Preference Page

This page allows the user to configure ACE's Multiprocess View (see page 204) to submit Multiprocess jobs to a third-party cloud/grid/job submission system, by 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. Be aware that the Oracle Grid Engine's qsub arguments are not 100% compatible with the qsub standard documented at http://pubs.opengroup.org/onlinepubs/9699919799/utilities/qsub.html.



When the third-party job submission system support is enabled, ACE will call 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) for more information.

Debugging job submission configurations:

If the job submission system is properly configured on the host machine, (meaning the user is able 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 the same command from the command-line.

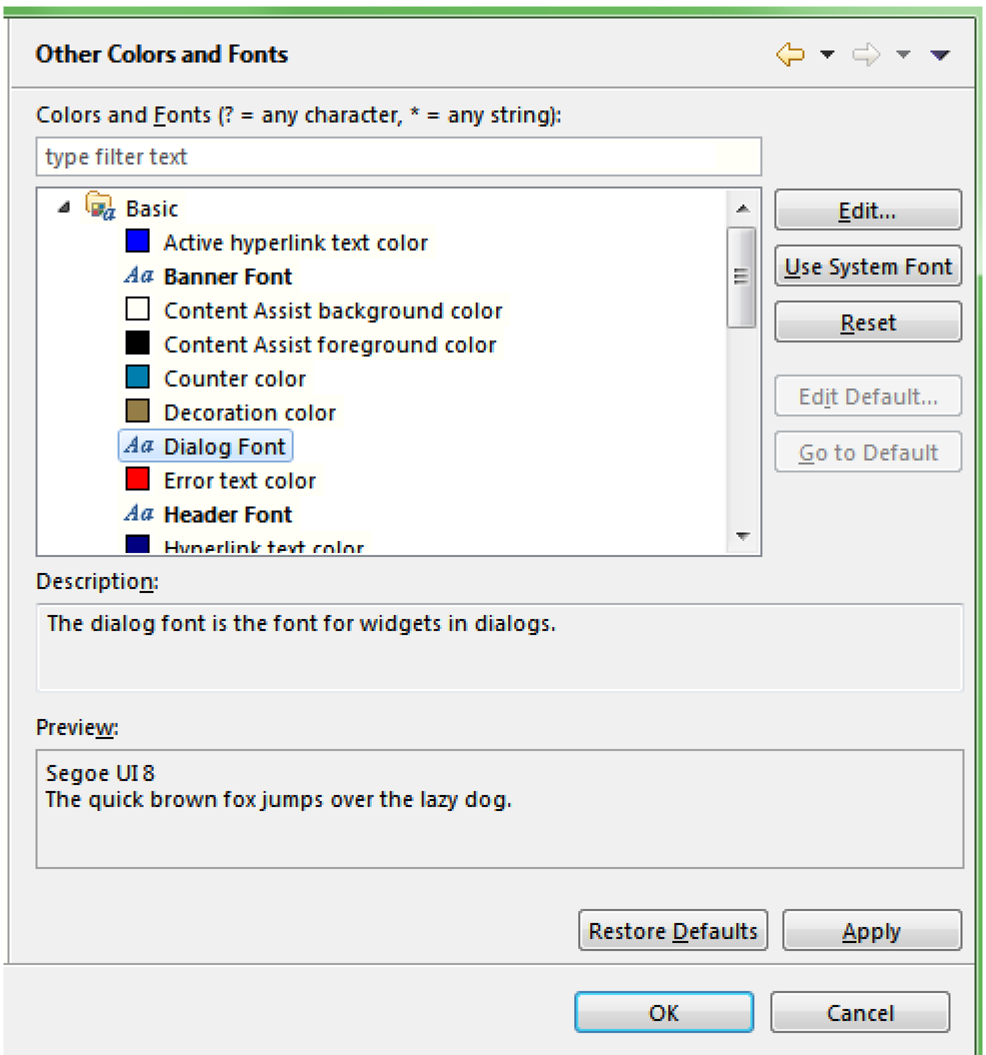
Other Colors and Fonts Preference Page

Many of the fonts and colors and used by ACE components can be set using the **General -> Appearance -> Other Colors and Fonts** to open the "Other Colors and Fonts" preference page. 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 choose the operating system font setting or by clicking **Edit** to open up a font selection dialog. **Reset** can be used to return to the default value.

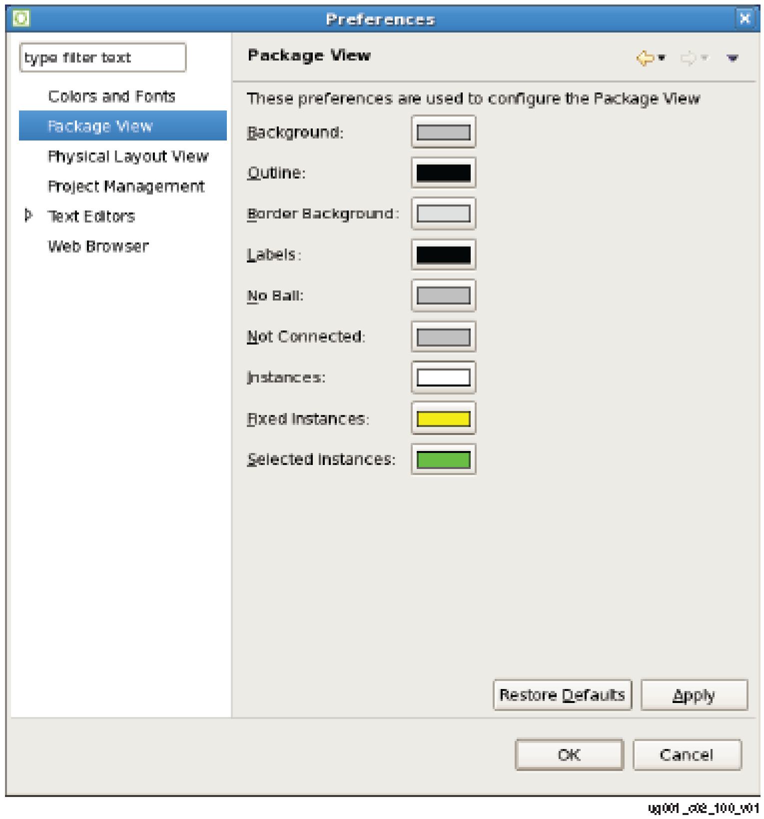
Color settings can be changed by clicking **Edit** to the right of the tree area when a color is selected. **Reset** can be used to return to the default value. The Colors and Fonts text field can be used to filter the contents. Simply type in an entry and any matching results remain in the tree view.

Descriptions and previews are provided when the Workbench colors and font settings are selected.



Package View Preference Page

View settings for the Package view (see page 226) are set via the Package View preference page. This page currently provides color configuration for several graphics layers in the Package view (see page 226) .



Placement Regions Preference Page

The Placement Regions Preference Page configures how Placement Regions (see page 435) are handled in the Placement Regions view (see page 234) and the Floorplanner view (see page 178) (when the Floorplanner's Placement Region Tool is active).

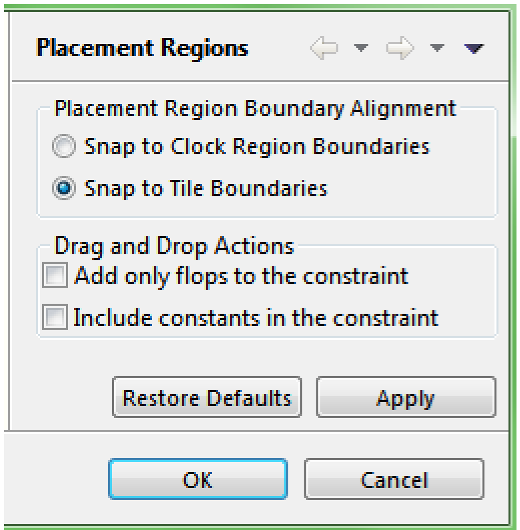
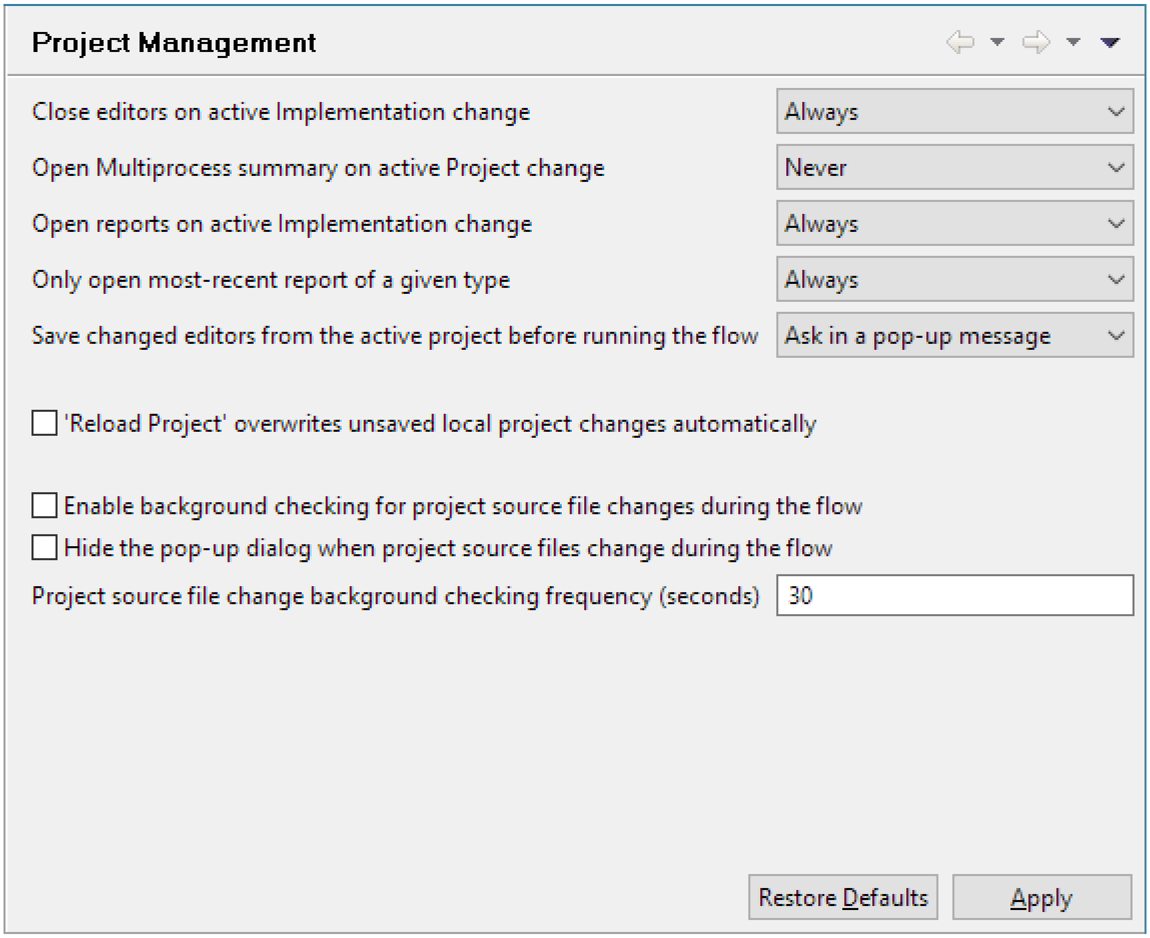


Table 152: *Placement Region Preferences*

Option		Description
Placement Region Boundary Alignment		
	Snap to Clock Region Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries will be forced to align with Clock Region Boundaries. Use this for a simple, coarse-grained approach to Placement Regions. Recommended setting for most use cases.
	Snap to Tile Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries will be forced to align with Tile Boundaries, for a very fine-grained region. Recommended only 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 will be assigned to the region constraint. All other dropped items will be excluded from the constraint.
	Include constants in the constraint	When using drag-and-drop to assign Placement Region Constraints, this setting allows constants to be included in the constraint. ⚠ This setting is ignored if "Add only flops to the constraint" is enabled.

Project Management Preference Page

The behavior of Editors (see page 25) and Reports (see page 322) is set from the Project Management Preference Page.



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 258).

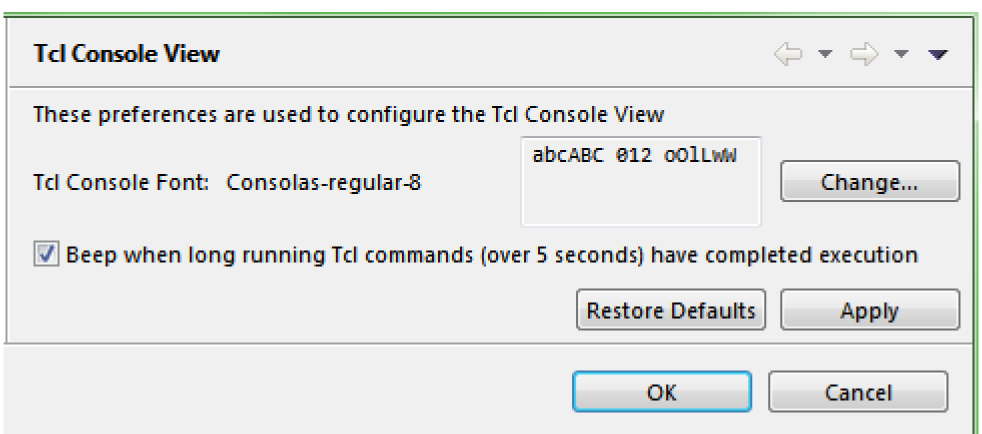


Table 153: *Tcl Console View Preferences*

Option	Description
Tcl Console Font	Allows the user to chose the font used in the Tcl Console.
Beep when long running Tcl commands (over 5 seconds) have completed execution	Enabling this will provide the user with audible feedback (the default system beep/bell sound) when long-running commands complete.

Text Editors Preference Page

The behavior and appearance of the text editor can be set from the Text Editors Preference Page.

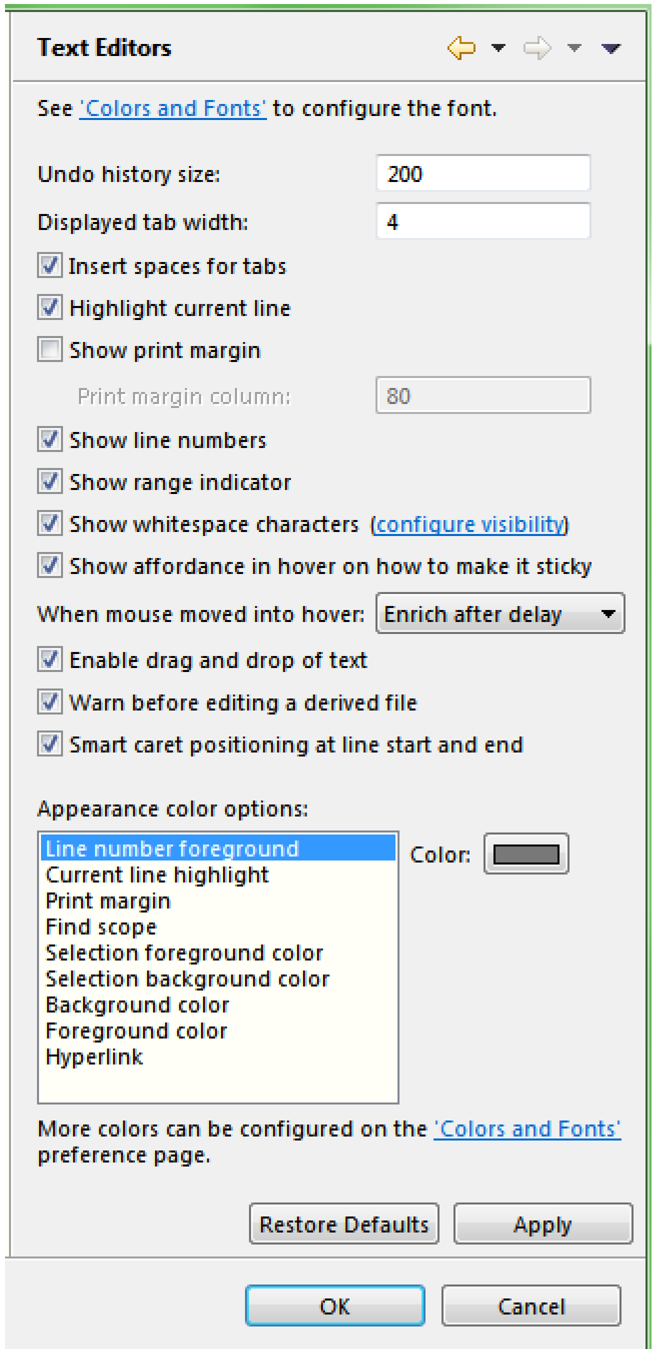


Table 154: *Text Editor Options*

Option	Default	Description
Undo history size	200	Sets the undo history size.
Display tab width	4	Sets the tab width for the editor.
Insert spaces for tabs	Deselected	Enables insertion of spaces for 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	Enables the visibility of the print margin.
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 preferences can be changed on the Quick Diff preference page, accessed via **Text Editors -> Quick Diff**.

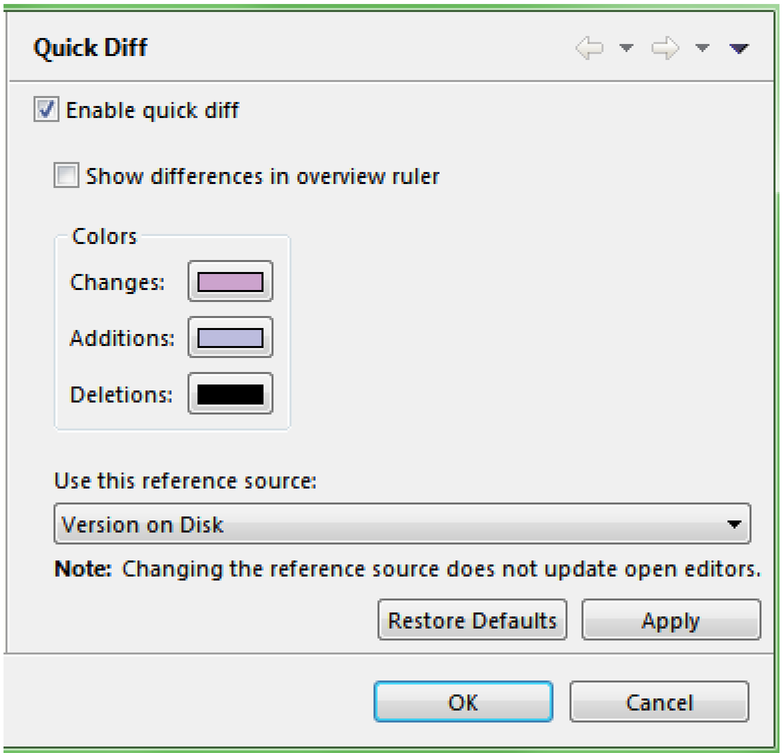


Table 155: *Quick Diff Preference Page*

Option	Default	Description
Enable quick diff	Selected	Enables/disables the quick diff option.
Show differences in overview ruler	Deselected	This option shows differences in the overview ruler.
Colors		
Changes		Sets the color indicating changes.‡
Additions		Sets the color indicating additions.‡
Deletions		Sets the color indicating deletions.‡
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.
‡ 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 312) may have multiple implementations. Each implementation contains the set of flow options (also called implementation options) configuring the project's run through the Flow (see page 316), 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 313) is located). Implementation directories are named with the implementation name and contain flow output files. Output Files (see page 315) 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 313). In the GUI, project implementations can be browsed in the Projects View (see page 238). Selecting an implementation activates (see page 316) it and displays its implementation options in the Options View (see page 215).

Once 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 358)). Implementations may later be restored from previously saved Acxdb files (See Restoring Implementations (see page 359)).

Implementation Options

There are a wide variety of configurable implementation options which will alter how ACE processes that implementation of the design as it moves through the flow (see page 316). The the most-commonly used option settings are displayed in the Options View (see page 215) for the current Active Project and Implementation (see page 316). Within the Options view, implementation options are grouped by flow steps (see page 317), to indicate which flow step the option affects. Changing the value of an option will cause that flow step's current results (if any) to become invalid/cleared, and that flow step (and all later flow steps) will need to be rerun, making use of the newly-changed option.

An Implementation Options Report (see page 325) of all available implementation options may be generated via the Tcl command report_impl_options (see page 575). 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 594), or reset back to the default values with reset_impl_option (see page 579).

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. These "option sets" are made available (with description) to users in the Multiprocess View (see page 204), 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 don't 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 Option 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's description are overwritten with the described values.

Achronix broke up the Option Sets into small granular chunks because of QOR/runtime tradeoffs. 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 323), users may choose to save hours of runtime if they only lose 0.01% frequency by using (for example) the Option Set "acx_util_pro" instead of "acx_all_opt" as they iterate their design. (Many of the Option Sets like "acx_util_pro" will enable a subset of what "acx_all_opt" enables.)

> Currently, the Option Set overrides will 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) will be identical.

It is expected that among all the Option Sets, users will be able to find at least one that provides the necessary QOR gain for an acceptable runtime impact, allowing the user the fastest possible design iteration.

See the Multiprocess View (see page 204) and Attempting Likely Optimizations Using Option Sets (see page 434) for more info.

Project File

Projects are persisted in project files (`.acxprj` file extension) 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. Users can edit project files manually and then load them into the tool to use as a script or for running regressions.

> When ACE loads a project file, it locks that file to prohibit other ACE sessions from loading the same file. This is done to prevent project data corruption, which could happen if two sessions tried 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 238). Project file contents may also be viewed in a Text Editor (see page 26) in the GUI by double-clicking on the project name in the Projects view (example file contents shown below):

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:

- Synthesized netlist files
- SDC/PDC/PRT constraints files

In the GUI, source files may be browsed in the Projects View (see page 238) and viewed in the built-in Text Editor (see page 26) by double clicking on 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 as follows:

File Type	Description
*.sdc, *.scf	Timing constraints files in SDC format. These files are read by the timer in ACE. The user must put all timing constraints in these file.
*.pdc	Placement constraints files for ACE. ACE can support many functions in this type of file, including placement and insertion of boundary pins. Users should not put any timing constraints 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 will be 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 25) (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 238) under the project's IP folder, and the associated editor may be started by double clicking on the file name in the Projects view.

For more details, see Creating an IP Configuration (see page 387), or one of the many IP Configuration Editors.

Output Files

Output files are generated for project implementations (see page 312) by running certain steps in the Flow (see page 316). By default, output files are automatically written to the project implementation's directory under the output or reports directory as appropriate. The user can also specify alternate output locations when running individual flow steps (see page 317) with command line options.

In the GUI, output files can be browsed in the Projects view (see page 238) under their implementation and viewed in the editor area by double clicking on 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.

When contacting Achronix technical support, users may sometimes be requested to provide one or more of these log files to Achronix to assist in the support effort.

ACE Session Log

Every time ACE is started, a new ACE session log is created in the directory <user_home_dir>/.achronix/. This file will be named ace_<date_timestamp>.log, where <*date_timestamp*> is *year_month_day_hour_minute_second*, with hour from a 24-hour clock. For example, if ACE was started in Linux with a username of "example_user", on January 11th of 2010 at 2:34:56pm, the complete log file name would be:

```
/home/example_user/.achronix/ace_2010_01_11_14_34_56.log
```

ACE session log messages are also sent to the Tcl Console View (see page 258). For more information, see Viewing the ACE Log File (see page 385).

Implementation Log

In addition to the session log, each project (see page 312) implementation (see page 312) has an implementation log maintained for the complete life of the implementation. All changes to the implementation, including running the Flow (see page 316) for that implementation, are appended to the implementation log. The implementation 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 204) do not have their log information appended to the ACE Session Log. This is because multiple processes would be appending info to the log file simultaneously, which would leave log entries interleaved in an unreadable mess. Instead, each implementation executed in the background will create a new log file named `multiprocess.log` in that implementation's log directory. (This will overwrite 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.

Snapshot Log

The Snapshot log file is discussed in Collecting Samples of the User Design (see page 427).

ACE GUI Log

On rare occasions, Achronix Tech Support may request the ACE GUI Log. This may be found by selecting: **Help -> About Ace -> Installation Details -> Configuration -> View Error Log**, which will open the GUI log in a non-ACE text file editor or HTML browser. The editor/browser will typically report 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 one 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's name, active implementation's 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 238)'s tree, the active project and its active implementation will both be 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 187). The current Flow Mode implementation option (selected in the Options View (see page 215)) will affect which Flow Steps may be executed.

A flow can only be run on a single Project (see page 312), and within that project a single Implementation (see page 312) at a time (these will be the Active Project and Implementation (see page 316)) during an interactive ACE session.

To run multiple implementations from the same project through the flow simultaneously, use the Multiprocess View (see page 204).

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 361).

Flow Steps

The Flow (see page 316) is composed of a series of flow steps, each representing a command operating on the design in the Active Project and Implementation (see page 316). 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 flow's default order of flow steps is displayed in the ACE GUI's Flow View (see page 187), with flow steps grouped into categories for organizational purposes.

The implementation option for Flow Mode (see page 321) (typically configured through the Options View (see page 215)) will affect which flow steps are able to be executed.

Table 156: *Achronix's Default Flow Steps and IDs*

Name		ID
Prepare		`prepare`
	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`

Name		ID
	Generate Final Simulation Netlist	`write_netlist_final`
FPGA Programming		`fpga_program`
	Generate Bitstream	`write_bitstream`
	FPGA Download	`fpga_download`

- All flow step IDs can be executed at the ACE GUI Tcl console (see Tcl Console View (see page 258)) or as part of the user Tcl script that can be invoked when running ACE (see page 344) 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 558).

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.

Once 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 done once the design is prepared (see Pre-Placing a Design (see page 399)).

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. Once 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. Once 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 317) under the Design Completion category will be skipped by default when the flow mode is set to **Evaluation**. See Flow Mode (see page 321) for more details.

Post-Process Design

After **Run Place** and **Run Route** have successfully completed (or an `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 your design fails final DRC checks, you can still generate a **Post-Route** timing report 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 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

> ⚠ The **Generate Bitstream** flow step will fail if attempted in **Evaluation** flow mode.
>
> Additionally, all Flow Steps (see page 317) under the FPGA Programming category will be skipped by default when the flow mode is set to **Evaluation**.
>
> See Flow Mode (see page 321) for more details.

Generate Bitstream

After a design is placed and routed, a bitstream for the target device can be generated. This step generates a bitstream (STAPL file) for the current implementation based on the settings in the Options view (see page 215) (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 will 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 the Bitporter pod 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 176) (see Playing a STAPL File (Programming a Device) (see page 432))

For more details, refer to the *Bitporter User Guide* (UG004).

Flow Status

Flow Steps (see page 317) each have a status associated with them. The current status of each flow step for the Active Project and Implementation (see page 316) can be seen in the GUI in the Flow View (see page 187). Each step can be in one of the following status states:

Table 157: *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 360) 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 316) and Flow View (see page 187), as well as the task for Running the Flow (see page 361) for more details.

Flow Mode

The flow mode is managed as an Implementation (see page 312) Option, typically through the Options View (see page 215).

The chosen flow mode will determine which DRCs are executed at different points in the Flow (see page 316), and in some configurations will prohibit the final Flow Steps (see page 317) from being able to execute successfully.

- **Evaluation** flow 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 users to more quickly iterate during preliminary or early design stages.
- **Normal** flow mode enforces all DRC checks prior to generating a bitstream. Some checks are flagged as warnings early on in the flow to give the user 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. This mode should be used when developing a real design for a product, and enables bitstream generation.
- **Strict** flow 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 187) will stop after the **Place and Route** flow step category is completed. By default, the flows steps under **Design Completion** and **FPGA Programming** will not be run unless the implementation is in **Normal** or **Strict** flow mode. Of particular note, the **Generate Bitstream** flow step will fail if attempted while in **Evaluation** flow mode.

> ⚠ Bitstream generation requires **Normal** or **Strict** flow mode, so that ACE may ensure all DRCs have passed.

Reports

ACE generates a number of reports to inform users how their designs are being handled in the selected Achronix device. These reports are meant to assist the user when 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 plaintext 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 317).

To generate this report manually, see the Tcl command `report_utilization`.

Pin Assignment Report

The Pin Assignment Report shows detailed information on each of the user design's top-level ports, including placement and configuration details.

Typically, the report is automatically generated by the Flow (see page 316) at multiple times (as part of the **Run Prepare**, **Run Place**, and **Post-Process Design** flow steps (see page 317)).

To generate this report manually, see the Tcl command `report_pins`.

It is also possible to generate the Pin Assignment Report from within the "IO Assignment View (see page 191)". This generated version also allows customization of the report's data contents, specifically the data columns used and the column ordering. For more information, see the "IO Assignment View (see page 191)".

Clock Report

The Clock Report shows all the 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 316) at multiple stages of the flow.

To generate this report manually, see the Tcl command `report_clocks`.

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 316), 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 317).

To generate this report manually, see the Tcl command `run_timing_analysis`.

If the user enables Timing Across All Temperature Corners (see page 331), a separate timing report will be generated for each temperature corner, with the file name of the report noting the corner being reported.

Report Content

The report will contain a Summary section and a Details section.

The Summary section will contain three tables. There will be 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 will contain a single row for each Clock/Group, showing the most critical path for that Clock/Group.

The Details section will contain a configurable maximum number of critical setup paths and critical hold paths for each Clock/Group, and each of those critical paths will include a configurable maximum number of worst paths for the critical path's endpoint.

The number of critical paths and worst paths are Implementation Options (see page) configured in the Options View (see page 215) 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 316) during the **Run Route** and **Post-Process Design** flow steps (see page 317).

To generate this report manually, see the Tcl command `report_routing`.

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 316) (and opened in the GUI) during the **Run Prepare** flow step (see page 317) when Using Incremental Compilation (Partitions) (see page 443).

To generate this report manually, see the Tcl command `report_partitions`.

Power Dissipation Report

The Power Dissipation Report shows various power-related statistics for the current design on the selected Achronix device.

This report is automatically generated by the Flow (see page 316) for eFPGA devices, but not FPGA devices.

This report can be generated on-demand using the Tcl command `report_power`.

If the user enables Timing Across All Temperature Corners (see page 331), a separate power dissipation report will be generated for each temperature corner, with the file name of the report noting the corner being reported.

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 316), but can be generated manually with the Tcl command `report_design_stats.`

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 317) are enabled), as well as process execution times, for selected Implementations (see page 312) of a single Project when executed from the Multiprocess View (see page 204). This report will be 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 312). While the Multiprocess flow is still executing, the report will contain incomplete results - rows containing incomplete data will be marked as such.

If errors are encountered during flow execution for one of the selected Implementations (see page 312), the row(s) of data for that implementation will be marked accordingly.

Because the Multiprocess Summary Report summarizes the results of multiple Implementations (see page 312) (unlike the other reports, which are generated for a single implementation), the HTML file containing the report is not placed in any implementation's `report` subfolder. Instead, this report is placed in the directory containing the `.acxprj` ACE Project File (see page 313), and is named `multiprocess_summary.html`.

> **If closed, the Multiprocess Summary Report can be re-opened at any time**
>
> To re-open the Multiprocess Summary Report for the active project at any time, 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 238).

This report is only available in HTML format. There is no Tcl command available to generate this report manually.

For more info on how this report is used, see Running Multiple Flows in Parallel (see page 363) and Attempting Likely Optimizations Using Option Sets (see page 434).

Timing Results Summary Section

The Timing Results Summary section of the report will only be generated if either the "Run Post-Route Timing Analysis" or "Run Sign-off Timing Analysis" Flow Steps (see page 317) are executed during the Multiprocess Flow. (Users must enable any desired optional flow steps in the Flow View (see page 187) before Multiprocess execution is started.)

While an implementation is waiting to be executed, or is currently executing, the Clock Domain column of the report will contain a message to indicate the implementation's execution status.

Once an implementation has completed execution, the timing results for that implementation will be populated in the Report. By default, each implementation will provide timing results (Upper-Limit Frequency, Worst Setup Slack, and Worst Hold Slack) for a single PVT combination (named in a column group header), though if the "Report all temperature corners" implementation option is enabled (see the Options View (see page 215)), multiple PVT combinations will be 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) will be 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 implementation's defined clock domain/PVT combinations. If the timing constraints were met for all clocks and all reported PVT combinations in an implementation, the Implementation Name cell will also be color coded green to indicate success. If errors are encountered during flow execution, the appropriate Implementation Name cell will be colored red, and "***(Flow Error)***" will appear in that row.

Hyperlinks to each detailed Timing Report (see page 322) (one is created for each implementation for each reported PVT) are made available under the first clock domain's Upper-Limit Frequency column for each PVT.

In some cases, some implementations will provide Sign-Off timing results but other implementations will not. This can happen most often when some implementations are in Evaluation Flow Mode (see page 321) 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 will indicate that the report contains Sign-Off data. All earlier Post-Route timing results will have a footnote indicating that they're not showing Sign-Off data.

Runtime Results Summary Section

This section of the report will always be generated, and will indicate the total process execution time and peak memory utilization for each implementation.

Rows in the table will indicate whether an implementation's flow execution is incomplete (running or waiting to run) or has encountered errors.

Users should be aware that implementation execution times will vary depending upon the available processors, available memory, and total load on the workstation executing ACE. An implementation's selected Implementation Options (see page) (and thus the Option Sets (see page)) can have a significant impact upon implementation runtimes. When using an external cloud/grid/batch Job Submission system, the reported times may include the time spent waiting in the external system's queues.

If comparing runtimes on multi-user workstations, please remain aware that workstation load may vary widely over the multiprocess execution period, making direct runtime comparisons difficult.

Even on single-user workstations, if using a **Parallel Queue Count** greater than one, be aware that the last-executed implementation(s) will likely 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, memory, I/O, etc.)

Because of all of these limitations, the reported implementation flow execution 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 316), but can be generated on demand using the Tcl command report_impl_options (see page 575).

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 215)). 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 215).

Note

Users should only alter the values of hidden options (those which are not shown in the GUI's Options View) 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 datamodel. 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 will not be unrouted by the run_unroute command, or when the user re-runs the run_route flow step 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 will be cloned, or buffer trees inserted, to keep each (non clock/reset) net's fanout under this limit. Applies only when the fanout_control impl option is enabled. Note that ACE will never insert 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, then the following can be used

Code

```
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 can't be deleted by any of the netlist optimization flow commands. Note that if the instance or net is logically redundant, it may 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 be done 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 will be placed together.

clock_type

Applies to nets or driver (output) pins.

Normally a user will set this property on a net or driver (output) pin using the set_clock_type TCL command in their 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 (see page 592) 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 that not all 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".

ace_useioff

Applies to ports, nets, IO 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 482) section.

ace_virtualize

Applies to ports only.

Allows the user to specify which ports and port busses will be virtualized when ACE is run in evaluation flow mode, and when the design contains more top level ports than available IO pads/pins. For more information see the Working with Virtual I/O (see page 490) section.

ace_virtualize_clock_port

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows the user to specify 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 490) section.

ace_virtualize_clock_net

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows the user to specify 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 490) 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 is used on the capture flop of user-supplied clock domain crossing synchronizer macros. For more information see the Speedcore IP Component Library User Guide.

LOCATION

Applies to instances only.

A string attribute giving the site name on which the instances 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 will all be specified by the chosen target device. A subset of this information is presented in the Clock Regions View (see page 167). 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 435) 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 will be used to color the Instance in the Floorplanner View (see page 178) and (for I/O Instances) the Package View (see page 226). 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 will be used to paint the instance.)

Additionally, several of the states have associated icons, which will normally be displayed alongside the instance when the instance appears in tables and lists, as in the Search View (see page 245), Selection View (see page 249), and Netlist Browser View (see page 211). Similar to the colors, the highest priority icon will be used. Thus, an instance that is both Fixed and Locked will use the Fixed icon.

Priority	State	Default Color	Icon	Description
1	Selected	bright green	-	The user has added this instance to the ACE Selection Set, typically either by using the Selection View (see page 249) or the Tcl select (see page 592) command. For more information, see Selecting Floorplanner Objects (see page 394).
				An instance that the user has chosen to Highlight, typically by using the Highlight Instance commands in one of the views within the Floorplanner Perspective, or by the Tcl highlight (see page 568) command. See Highlighting Objects in the Floorplanner View (see page 396) for more information.

Priority	State	Default Color	Icon	Description
2	Highlighted	(user-defined)	-	Highlights are not currently supported in the Package View.
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 the user has marked as "fixed". As long as an instance is defined with hard fixed placement, ACE will not change the site assignment for that instance during the Placement phase of PnR. For more information, see Placing an Object (see page 399).
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 will not change the site assignment for that instance during the Placement phase of PnR. For more information, see Using Incremental Compilation (Partitions) (see page 443).
6	Default (Soft) Placement	dark grey		A placed instance with no other specially defined state will be displayed in this manner.
7	Unplaced	-		An instance without a current site assignment (placement).

The colors mentioned are the default colors. These colors may be modified on the Floorplanner View Colors and Layers Preference Page (see page 292) and Package View Preference Page (see page 305). On the same preference pages, the user may toggle whether some states are shown at all, and the user may partly alter the render priority of some states.

Filter Properties

Several Tcl commands [find (see page 552), filter (see page 551)] allow Tcl command-line filtering of object lists. Additionally, the Search Filter Builder Dialog (see page 285) performs a similar function for the Search View (see page 245) in the ACE GUI.

The allowed filtering properties, operators, and values (where applicable) are as follows:

Table 158: *Supported Filter Properties*

Property Name	Operators	Values	Description
@attribute	=	*property*:*value*	The @attribute filter allows you to filter 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 (ie. prop:value).

Property Name	Operators	Values	Description
@clock_domain	=	*clockDomainName*	The @clock_domain filter allows you to filter instances, nets, paths, and pins based on clock domain name. Note that some instances may be part of multiple clock domains, such as a CDC instance.
@direction	=	in, out, inout	The @direction filter allows you to filter ports and pins based on direction.
@driver_type	=	(device-dependent)	The @driver_type filter allows you to filter nets and pins based on the type (cell name) of the driving instance.
@driving_net	=	*netName*	The @driving_net filter allows you to filter instances based on a net name that is driving them.
@driving_pin	=	*pinName*	The @driving_pin filter allows you to filter instances and nets based on the name of a pin that is driving them.
@fanout	=, <, >	(integers > 0)	The @fanout filter allows you to filter 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 you to filter instances based on whether placement is fixed or not.
@partition	=	*partitionName*	The @partition filter allows you to filter instances, nets, paths, and pins based on partition name.
@placed	=	true, false	The @placed filter allows you to filter instances, nets, and pins based on whether they are placed or not. In this context, a "placed" net means the net is routed.
@power	=, <, >	(floating point numbers > 0.0)	The @power filter allows you to filter 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 you to filter nets based on level of power consumption relative to other nets. The most power-consuming net will be ranked 1, the least power consuming net will be ranked n. Valid @power_rank values must be integers greater than or equal to 0.
@region	=	*placementRegionName*	The @region filter allows you to filter instances, nets, paths, and pins based on placement region name.
@sink_type	=	(device-dependent)	The @sink_type filter allows you to filter nets and pins based on what type (cell name) of instance(s) the net is driving.
@type	=	(device-dependent)	The @type filter allows you to filter instances based on type (cell name) of 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**
>
> ACE only optimizes place and route for a single user-chosen temperature corner per implementation, but can then report the timing analysis results of that routed netlist for all temperatures of interest. If a user wishes to optimize place and route for multiple temperatures, each desired temperature must be configured as a separate Implementation (see page 312). In other words, ACE cannot optimize a single netlist against all temperature corners simultaneously.

ACE requires the user to set the desired junction temperature for which the design needs to be placed and routed for the target F_{MAX}. ACE then uses this selected junction temperature to load the corresponding timing libraries and then optimizes the place and route to close timing at the requested frequency F_{MAX}. This junction temperature target for optimizing place and route can be set within the Options View (see page 215) under **Design Preparation → Junction Temperature**. With the desired junction temperature selected, ACE will place and route the design and generate the final timing report for this chosen junction temperature (which is one of the temperature corners for which the design needs to close timing at).

Now, if the user wishes to ensure that this place-and-route result is able to close timing and find out the achievable F_{MAX} at other temperature corners (other than the selected junction temperature), ACE provides an additional setting in ACE to generate timing reports at other temperature corners of interest. This option can be enabled in the Options View (see page 215) under **Timing Analysis → Report all temperature corners** (refer to the table of Timing Analysis Implementation Options under Options View (see page 215)).

> **Note**
>
> When the option **Report all temperature corners** is enabled, both the Timing Report (see page 322)s and the Power Dissipation Report (see page 323)s will be generated for each temperature corner.

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, the user has still has an additional opportunity to apply different optimization strategies by running the design through a Multiprocess run via the Multiprocess View (see page 204) GUI. With the **Report all temperature corners** option enabled when running 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 out all timing results across all temperature corners.

Here are some snapshots of the Multiprocess Summary Report (see page 323) for the same design at different junction temperatures with the **Report all temperature corners** option enabled.

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 other things.

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 which the net connects to must be placed (their respective site pins do not need to be placed).

When a net is *legal*, the router is able to route the net. An *illegal* net will cause 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 will inform the user about illegal nets caused by each ECO command, including why such nets were deemed illegal. It is up to the user to keep track of such nets and fix 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 317). 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 the design will refuse to perform net-routing with nets connected to unplaced instances. If the user so chooses, they may opt to 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 will be 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., they lost a pin connection that kept them legal). In these circumstances, a warning is printed to the Tcl console. The user must identify and resolve all such illegal nets before they can proceed.

Once any illegal nets have been made legal (driver added to floating net, sink pin added to dangling net, drivers removed from multi-driven net), the user must call `rewire_net <netName> -reroute` to physically create the connections specified in the ECO-modified netlist.

If the user decides to perform `run_prepare` after performing ECO, all changes made will be overwritten as `run_prepare` sources the original netlist(s) specified in the ACE project, but not any ECO modifications. As such, the user must be aware that they may undo all their work accidentally 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 will report 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 the user to 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]
```

This command deletes the named instances and disconnect the instances' pins from their respective nets. Any nets left with no connections are not automatically deleted. The user must delete these nets manually using `ace_eco::delete_net`.

Arg Name	Optional	Description
<instances>		List of user design instances to be deleted. The "i:" prefix is optional on each instance.
[-reroute]	✔	The optional -reroute option 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> ...}
```

This command deletes the named nets.

Arg Name	Optional	Description
<nets>		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> ...}
```

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} ...} ...
```

Arg Name	Optional	Description
<instances>		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]
```

This 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).

Arg Name	Optional	Description
<instance_name>		The name which will be given to the newly inserted instance. The "i:" prefix is optional.
<cell_type_name>		Type of instance. This must be a valid cell type for the current fabric.
-site	✔	The optional -site option names the site where the new instance will be placed. The site named must be compatible with the cell type, or the command will abort before the instance is created. The "s:" prefix is optional.

Arg Name	Optional	Description
-pins	✓	The optional -pins option specifies a list of pin name/net name pairs. Each named pin will be connected to the associated named net. The "p:" and "n:" prefixes are optional.
-parameters	✓	The optional -parameters option specifies a list of user parameters and values for the new instance. The parameters must be compatible with the cell type, or the command will abort before the instance is created.
-fixed	✓	The optional -fixed option indicates that given a -site parameter, the newly created instance will be fixed to that site. By default, Soft Placement will be performed (no fixing).
-reroute	✓	The optional -reroute option 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 will refuse 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]
```

This 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.

Arg Name	Optional	Description
<net_name>		The name which will be given to the newly inserted net. The "n:" prefix is optional.
<pins>		List of fully-qualified (including instance) pin names. Each named pin will be connected to the new net. The "t:" pin prefixes are optional.
-route	✓	The optional -reroute option 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 will not connect to pins that are already connected to a net; the user must disconnect those pins 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 will not be 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]
```

ThIS command enables the user to connect or disconnect an instance's pins 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.

Arg Name	Optional	Description
<instance_name>		The name of the instance whose wiring will be changed. The "i:" prefix is optional.
<pin_net_pairs>		List of user design pins/ports paired with the optional nets to which each will be connected. The "p:" and "n:" prefixes are optional. Example: "{pin1 net1} {disconnected_pin2}"
-reroute	✓	The optional -reroute option is used to re-route nets after the instance has been inserted.
-disconnect	✓	The option -disconnect option causes all existing connections to be disconnected before applying new pin/net connections.

> **Note**
>
> If a named pin/net connection already exists as specified, then that argument will be safely ignored.
> If a pin name is provided without a net name, that pin will be 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 the user to connect/disconnect 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 the user must know to route partially/unrouted nets left behind by their work.

Arg Name	Optional	Description
<net_name>		The name of the net whose wiring will be changed. The "n:" prefix is optional.
-connect	✓	The optional -connect option specifies a list of pins to be connected to net. If already connection, they will first be disconnected from their original nets.
-disconnect	✓	The optional -disconnect option specifies a list of pins that should be disconnected from net.
-clocktype	✓	The optional -clocktype option indicates the clock type of the pins to be connected.
-reroute	✓	The optional -reroute option is used to re-route nets after the instance has been inserted.
-verbose	✓	The optional -verbose option generates additional feedback as the command is running.

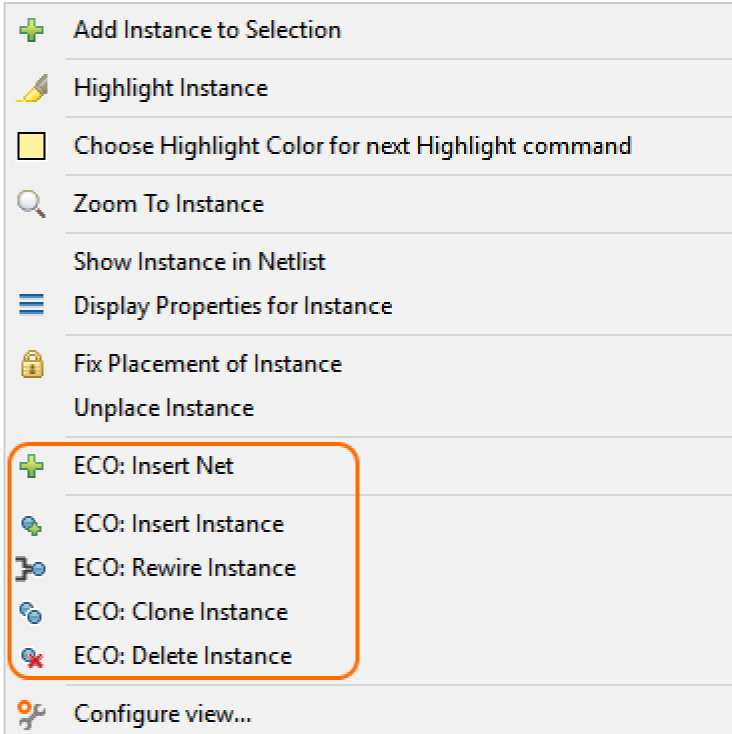
GUI Support

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 will appear in right-click context menus available on most views in the Floorplanner Perspective. For example, right-click on a net to display the available ECO net commands; right-click on an instance to display available ECO instance commands, etc.

Warning!

The GUI's 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 258) as the ACE ECO Tcl commands themselves are executed by the wizards.



Add Instance Pin Dialog

The Add Instance Pin dialog is used to choose an existing pin from an existing instance.

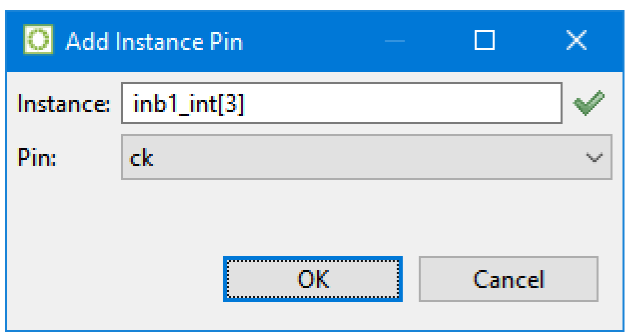


Table 159: *Add Instance Pin Dialog Fields*

Field	Description
Instance	A valid instance name. If an invalid name is specified, the green check mark next to the Instance field will change to indicate the error.
Pin	A list of the instance's pins.

ECO Insert Instance Dialog

The ECO: Insert Instance dialog allows the addition of a new instance.

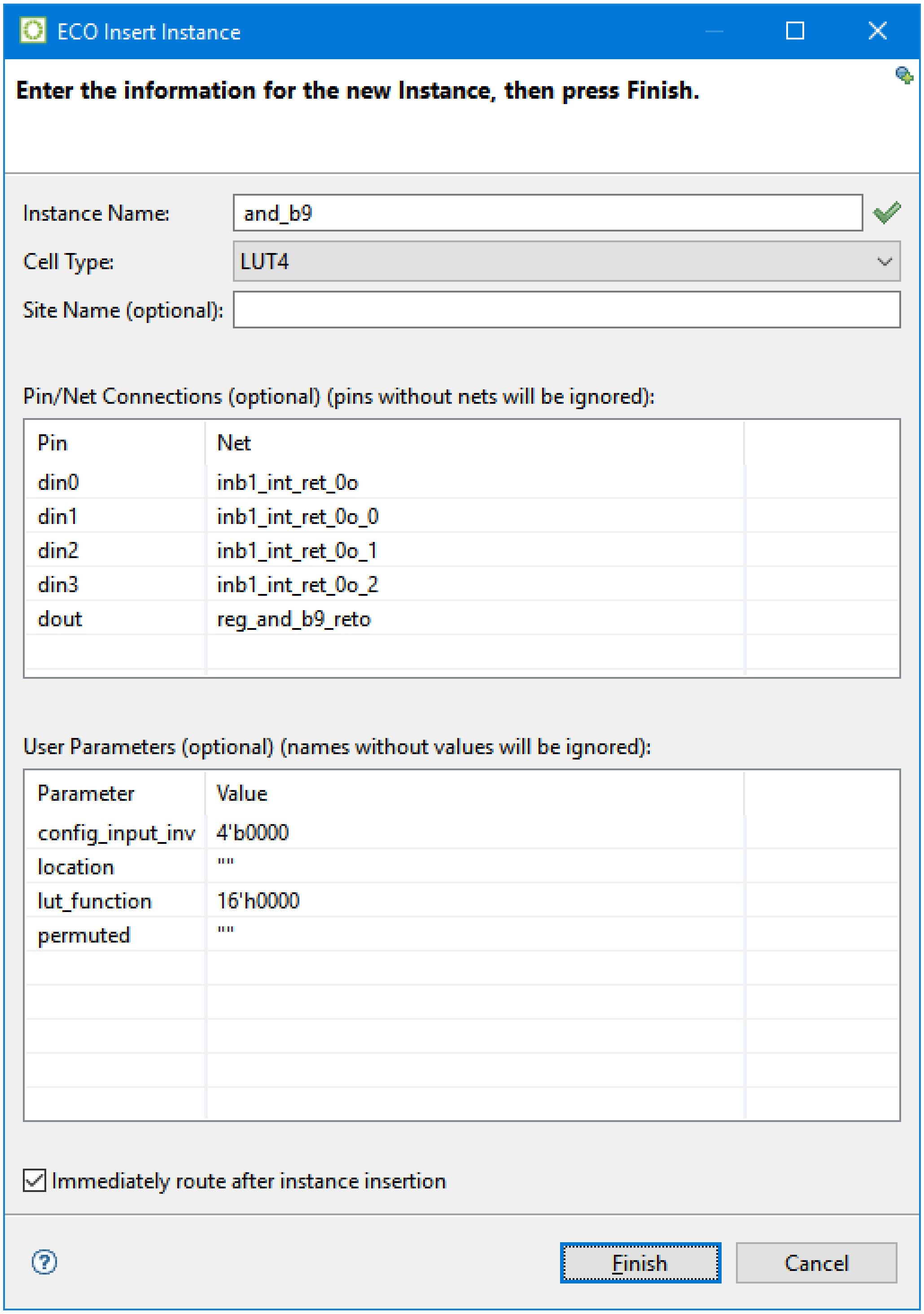

ECO Insert Instance
Enter the information for the new Instance, then press Finish.
Instance Name:
and_b9
Cell Type:
LUT4
Site Name (optional):
Pin/Net Connections (optional) (pins without nets will be ignored):
Pin
Net
din0
inb1_int_ret_0o
din1
inb1_int_ret_0o_0
din2
inb1_int_ret_0o_1
din3
inb1_int_ret_0o_2
dout
reg_and_b9_reto
User Parameters (optional) (names without values will be ignored):
Parameter
Value
config_input_inv
4'b0000
location
""
lut_function
16'h0000
permuted
""
Immediately route after instance insertion
Finish
Cancel

Table 160: *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 will change to indicate the error.
Cell Type	The cell type for the new instance.
Site Name (optional)	The name of the site where the new instance should be placed.
Pin/Net Connections	Click on a cell in the Net column to connect a pin to an existing net.
User Parameters (optional)	Click on a cell in the Value column to specify a parameter value. Parameters without values will be ignored.
Immediately route after instance insertion	If enabled, the design will be rerouted immediately after the new instance is inserted.

ECO Insert Net Dialog

The ECO: Insert Net dialog allows the addition of a new net.

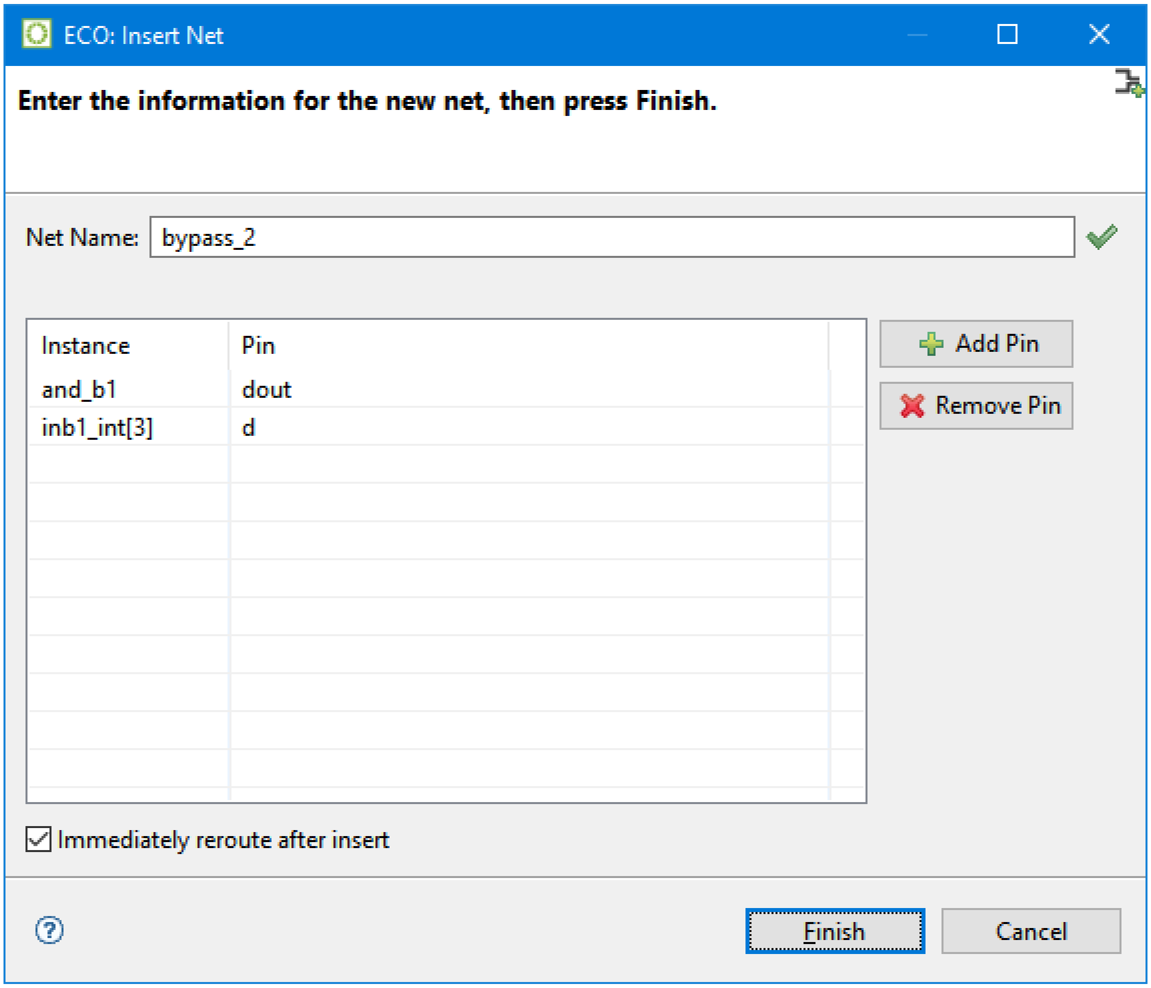


Table 161: *ECO Insert Net Dialog Fields*

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 will change to indicate the error.
Add Pin	The Instance Pins table in the dialog lists the pins that the new net will be connected to. 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 will be rerouted immediately after the new net is inserted.

ECO Rewire Instance Dialog

The ECO: Rewire Instance dialog allows an existing instance's properties to be adjusted.

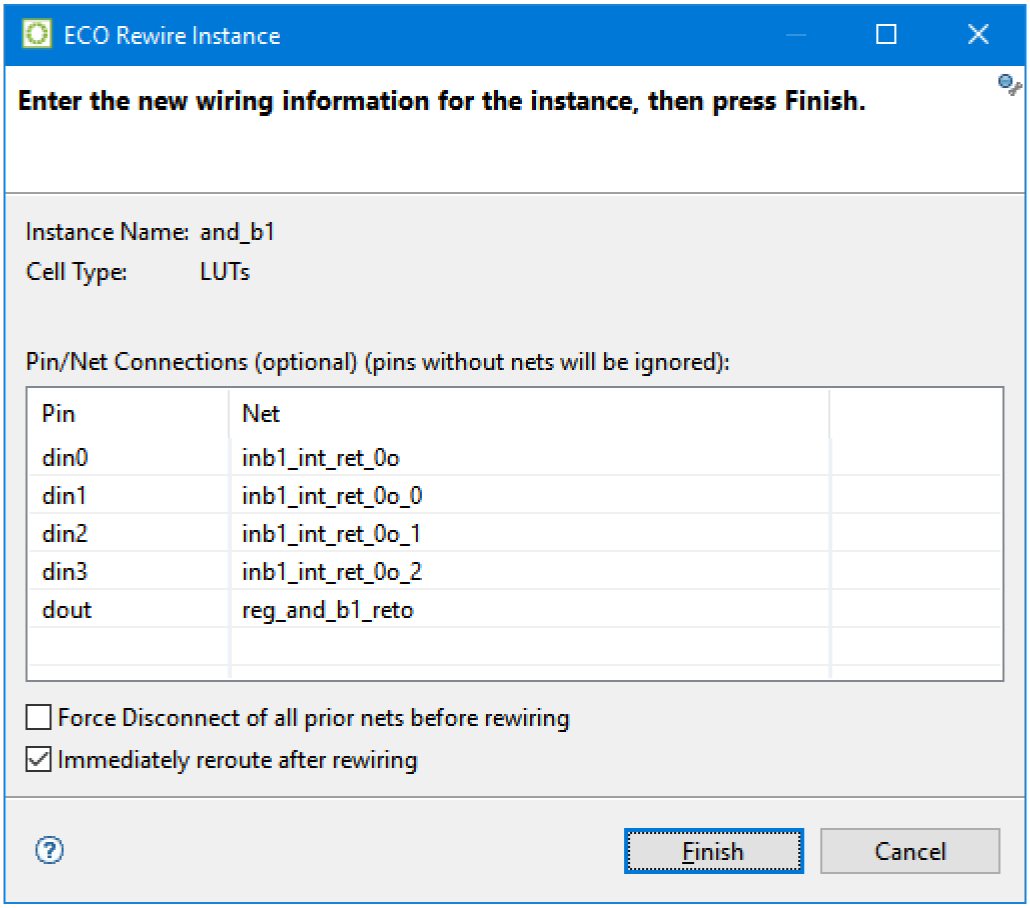


Table 162: *ECO Rewire Instance Dialog Fields*

Field	Description
Pin/Net Connections	Click on 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.
Immediately route after instance insertion	If enabled, the design will be 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.

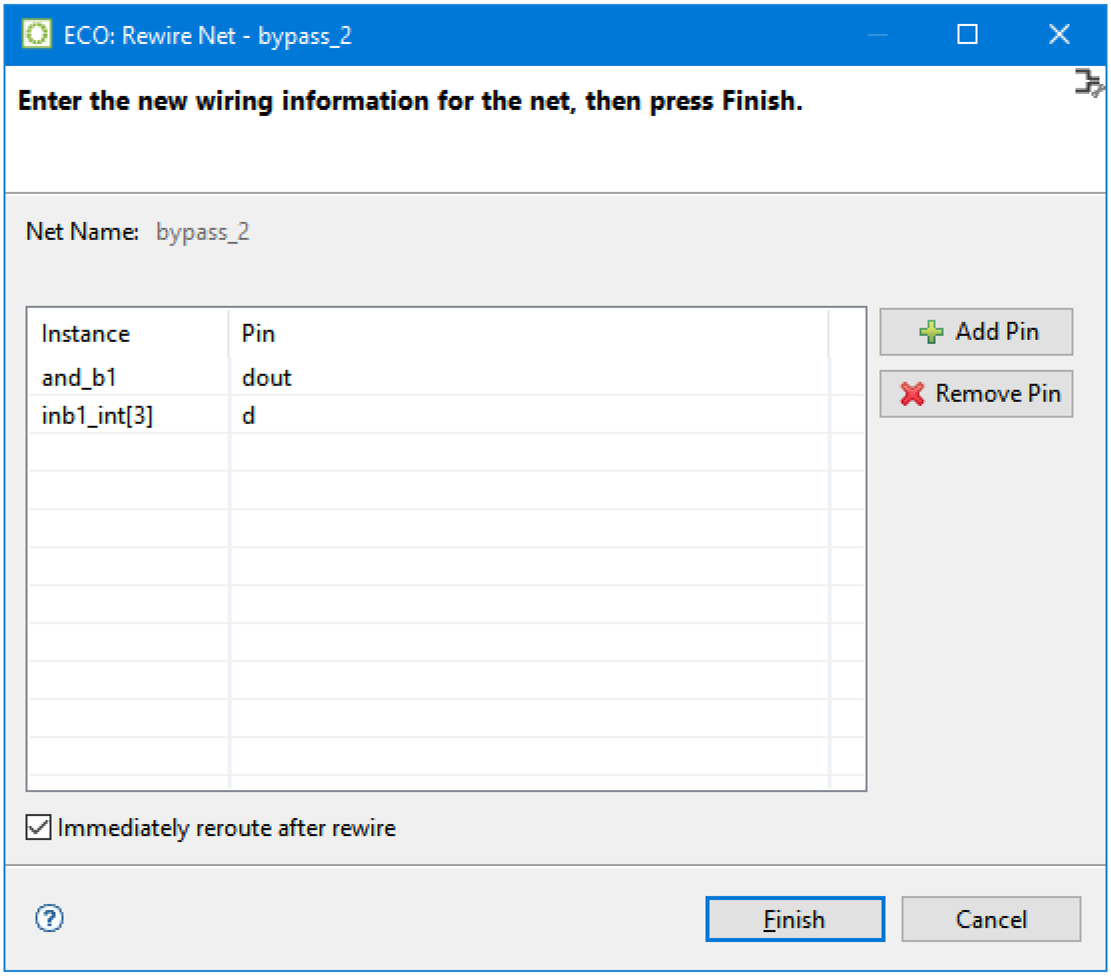


Table 163: *ECO Rewire Net Dialog Fields*

Field	Description
Add Pin	The Instance Pins table in the dialog lists the pins that the new net will be connected to. 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 will be rerouted immediately after the Insert Net dialog is completed.

Chapter - 3: Tasks

While the Concepts (see page 23) section was primarily concerned with which features exist in ACE, this Tasks section is concerned with how users may best utilize the features in ACE.

Running ACE

ACE can be run with full functionality in three different modes:

- GUI Mode (see page 344)
- Command-line Mode (see page 344)
- Batch Mode (see page 345)

A fourth mode, Lab Mode (see page 345), is also available, with reduced functionality.

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 the user to interactively enter 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)

The ACE GUI 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 solely chip programming and debugging. In this mode, a license is not required, but Tcl functionality will be unavailable, and the user is unable to work with their project files, run PnR, view the Floorplanner, configure IP, etc.

When the GUI is in this mode, only the views within the Bitporter Perspective and HW Demo Perspective are usable. The views within the Project Perspective, Floorplanner Perspective, and IP Configuration Perspective will be non-functional, since they require Tcl functionality.

Starting ACE in Lab Mode

```
% ./ace -lab_mode
```

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 the icons on the main toolbar.

It is also possible to see the choices of available choices as a menu, visible at **Window** → **Open Perspective**.

Descriptions of the available perspectives are found in the section describing the Perspectives (see page 23) concept.

Resetting Perspectives

Often, when users are altering positions of Editors (see page 25) and Views (see page 163) within a perspective, a user may end up with an arrangement they no longer find appealing. Rather than try to manually move the Views and Editors back to the original positions, it can be much faster and simpler to just reset the perspective.

To restore a perspective to its original layout, select **Window -> Reset Perspective** on the menu bar and click **OK** on the pop-up 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, and 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 main 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) will dock if the left mouse button is released at the current mouse location. Drag the tab near the left, right, top, or bottom border of another view or editor to see how that view/editor will split its available area with the dragged tab. Drag the tab near the tabs of another tab stack to see where the dragged tab will 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 location desired relative to the view or editor area underneath the mouse, release the left mouse button.

Table 164: *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 347) view/editor will 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 will occur.

> **Caution!**
>
> There is currently a known bug (***Linux-only***) in the application frameworks underlying ACE that may cause view /editor tab movements to detach (see page 347) instead of docking when the Help Window is open. See the Troubleshooting (see page 602) 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 on the tab of the view/editor to be moved and drag it to where it is desired. 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. The tab is now moved.

> **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 mouse button down) the tab of the view/editor to be detach.
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 be shown 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. Holding down the left mouse button, drag that editor 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 when the mouse button is released.
3. (Optional) 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 consisting of (in its basic form) an Editor Area (containing one or more stacks showing the open editors) surrounded by one or more View Stacks (each containing one or more views). These various parts compete for valuable screen real-estate, and correctly managing the amount of screen given to each can greatly enhance your productivity within ACE.

The two most common mechanisms for managing this issue are 'minimize' (i.e. make it use as little space as possible) and 'maximize' (i.e. give it as much space as possible). ACE provides a couple ways to access these operations:

1. Using the minimize and maximize buttons provided on a stack's border
2. Double-clicking on 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/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 178) 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 any open views in the perspective by using the icons in their Trim Stack (the area around the edges of the window is called the 'trim').

Editor maximization operates on a complete Editor Area (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 will cause it to be moved into the trim area at the edges of the workbench window, creating a *Trim Stack*.

> Be aware that the first time a stack is minimized, the Trim Stack may end up on any edge of the window. If the user manually moves the Trim Stack to a particular window edge, that same edge will typically be reused when that stack is re-minimized.

View Stacks get minimized into a trim representation that contains the icons for each view in the stack:

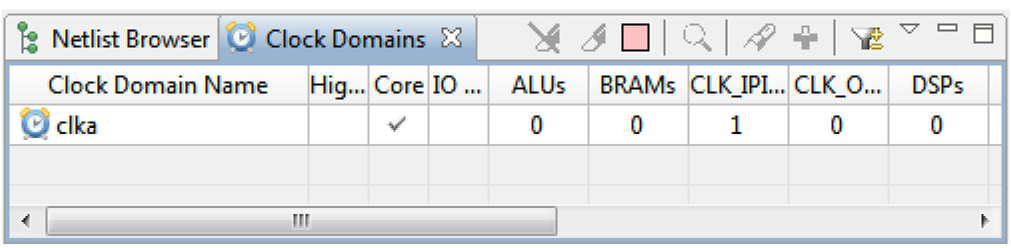


Figure 36: ***Example View Stack before minimization***

Figure 37: ***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 the icons would be the same, making them essentially useless).

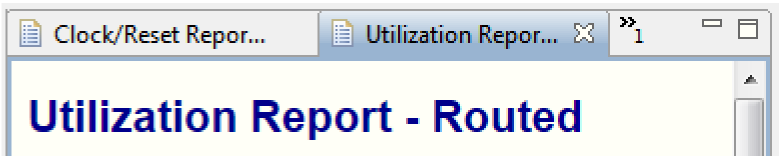


Figure 38: ***Example of Editor Area before minimization***

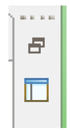


Figure 39: ***Example of minimized Editor Area Trim Stack***

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,) users can still gain additional screen space by minimizing the stacks that aren't of current interest. This will remove them from the main presentation and place them on the outer edge of the workbench window as *Trim Stacks*, allowing more space for the remaining stacks in the window.

There are two ways to end up with a stack in the trim:

- Directly minimizing the stack
- As the result of another stack being maximized

Depending on how the Trim Stack was created, its behavior is different: when un-maximizing (restoring from a maximized state), only those trim stacks that were created (automatically minimized) during the initial maximize will be restored to the main presentation, while stacks that were independently (manually) minimized will stay minimized.

This difference is important in that it allows users fine-grained control over the presentation. While using maximize is a one-click operation, it's 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, users may find themselves wanting to have the presentation show more than one stack. In these scenarios users shouldn't maximize; they should instead minimize all the other stacks *except* the ones wanted in the presentation. Once it is set up, users can still subsequently maximize the editor area, but the subsequent un-maximize will only restore the particular stack (s) that were sharing the presentation, not the ones explicitly/manually minimized.

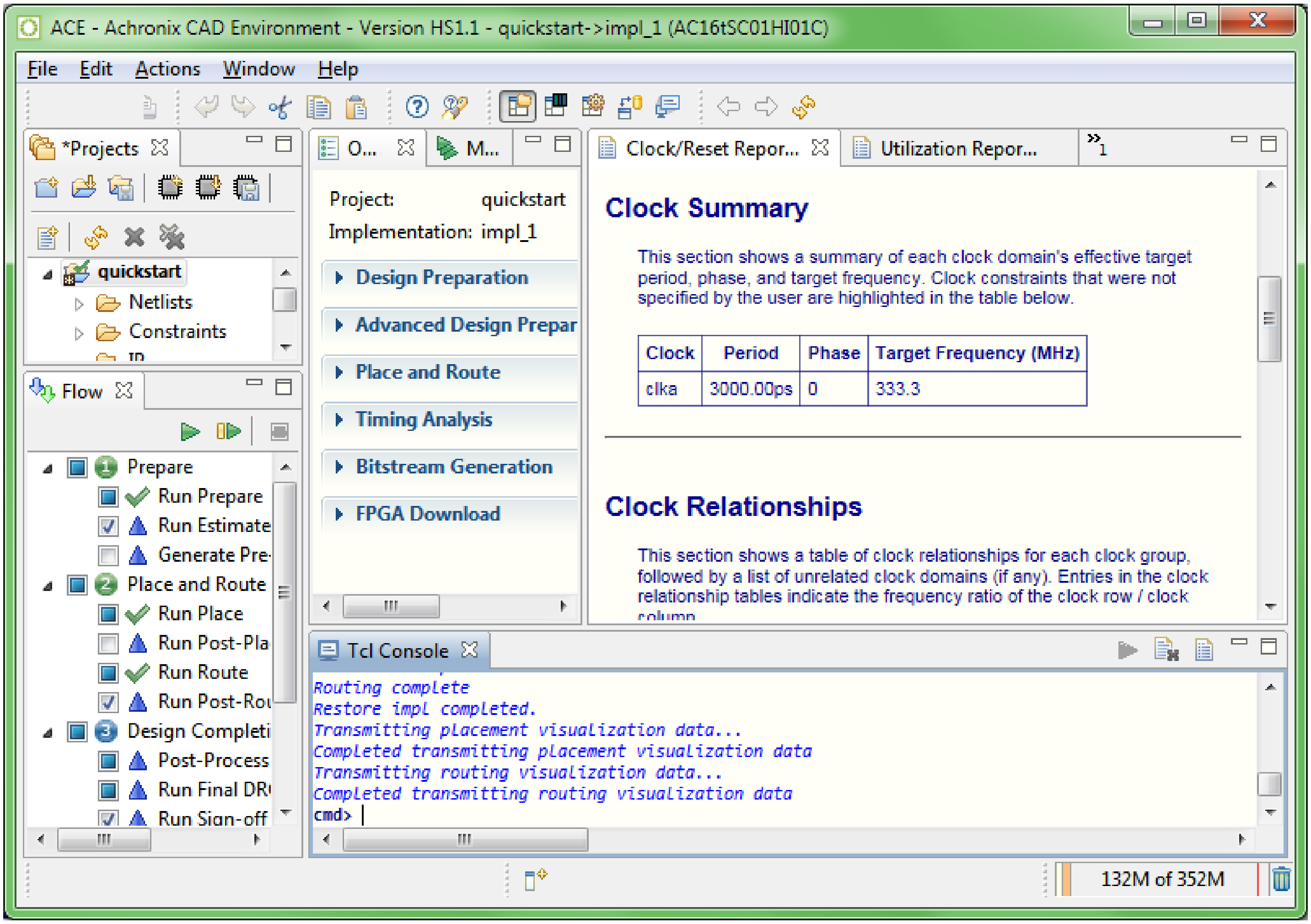


Figure 40: ***Example default presentation of the Projects Perspective***

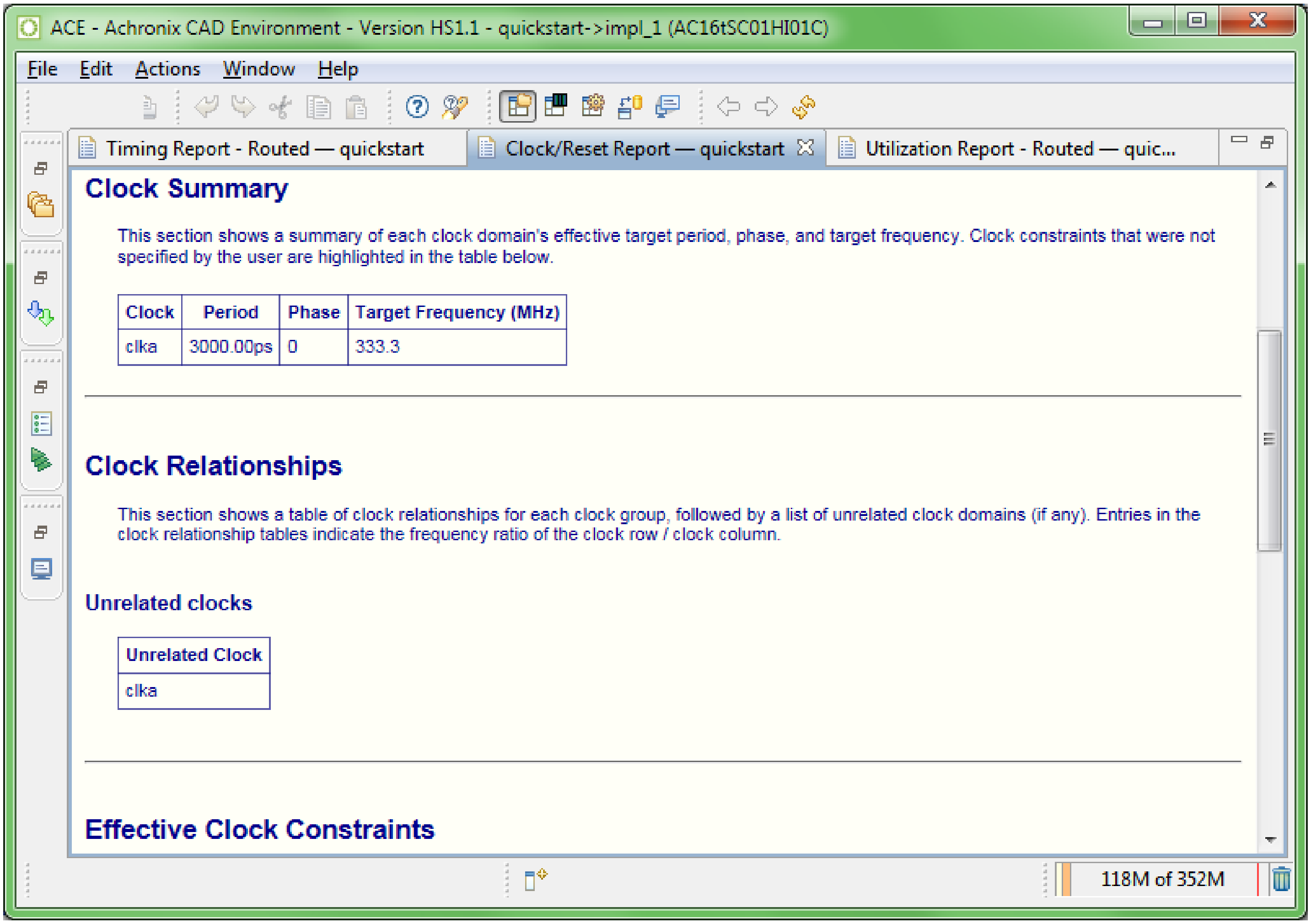


Figure 41: ***Example of the Projects Perspective with the Editor Area maximized***

Working with Projects and Implementations

Creating Projects

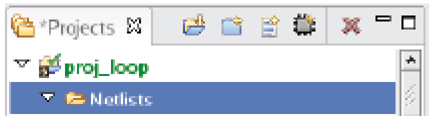
To create a new project in the workspace:

1. Click on 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.
   Note: *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 now 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.



ug001_c08_18_012108

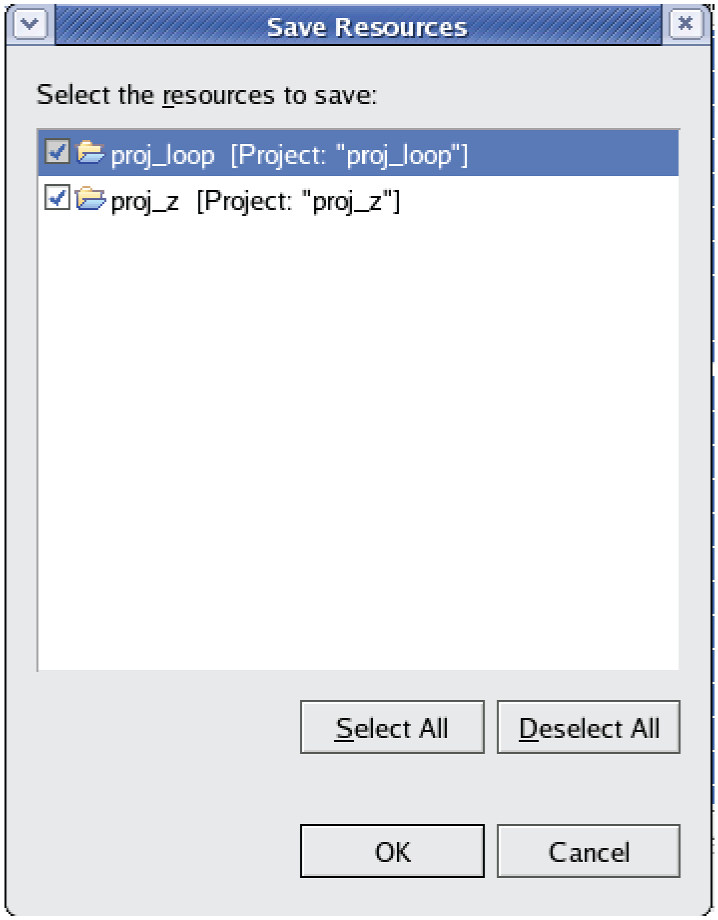
To save the changes to a project:

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.



ug001_c08_14_012103

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 will not be automatically re-loaded, nor will any related error be reported. Additionally, project files from the prior session which are no longer found in the file system will not be loaded, nor will any related errors be reported.

Loading a Project Using the GUI

To load existing Projects (see page 312) into the workspace:

1. Click on the **Load Project** toolbar button ( ) in the Projects View (see page 238).
2. In the Load Project Dialog (see page 276), **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 569) 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 359) for details on loading a prior place-and-route state.

> **Default Implementation Options Change Over Time**
>
> The default Implementation Options (see page) for ACE change over time as new optimizations become available and existing optimizations are refined. When a user loads a project from an earlier version of ACE, the user is shown a "Project Version Mismatch" popup dialog offering to reset all Implementation Options of all implementations to the latest default values.
>
> If the user is hesitant to risk losing old optimizations saved in implementation options, they may say no to the offered reset. If the user still wants to see how the new default implementation options could affect their design, they may simply create a new implementation for their project. The new implementation will contain all the new default values for implementation options, but will contain no place-and-route data. Be aware that since no constraint files from the project are enabled by default in a new implementation, the user will need 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 569) and restore_project (see page 580) may be used to open projects (and potentially also the project's most recent implementation) in ACE:

- The `load_project` command is simple, and will only open 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

> **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 will mention the username and hostname of the session which created the project lock, allowing users to coordinate sequential (not simultaneous!) project access.

Example error message for 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, users needing simultaneous access to a design should consult their Achronix FAE. Some potential options include Using Incremental Compilation (Partitions) (see page 443), or using version control tools to store the project.

> **Caution!**
>
> While 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 will 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 604)" under Troubleshooting (see page 602).

Removing Projects

To remove a project from the workspace:

1. Select a project in the Projects view.
2. Click on 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 user's responsibility to clean up unwanted files on disk. The project is left untouched on disk so that the project may be loaded again later if desired.

Opening Project Files in an Editor

To open a project file in the editor area, double-click on 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 312) in the Projects View (see page 238) to which the source files will be added.
2. Click on the **Add Source Files** toolbar button ( ) in the Projects view.
3. In the Add Source Files Dialog (see page 261), browse to the location of the source files.
4. Select the desired file in the dialog, and press **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's Netlists and/or Constraints nodes in the Projects View (see page 238).

After pressing **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.

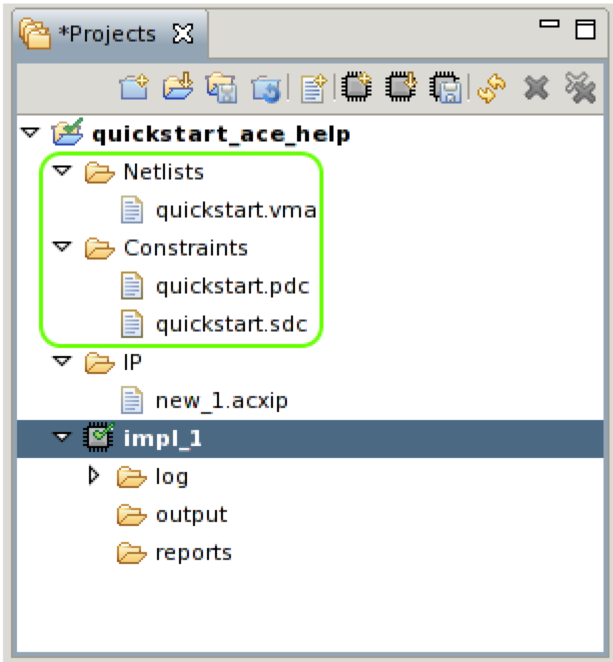
See also: add_project_netlist (see page 536), add_project_constraints (see page 535), add_project_ip (see page 535), Removing Source Files (see page 357), enable_project_constraints (see page 550), disable_project_constraints. (see page 545)

Source File Load Order

Note

Source file load order is shared by all Implementations (see page 312) within a given Project (see page 312). Enablement 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 215).

To assist the user's understanding of the load order of the source files, the netlist and constraint files are listed in the Projects View (see page 238) in the same order in which they will be loaded (the constraint files are additionally listed in order within the Options View (see page 215)).



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 happen in source file `create_clocks.sdc`, while operations upon that created clock may happen in source file `alternate_clocks.sdc`. To avoid errors, the user should first add `create_clocks.sdc` to the project as a source file, then add `alternate_clocks.sdc` as a source file. If the user tries to add all the files to the project in a single operation, the results are platform dependent, but often the operating system will "helpfully" sort 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, users have a few ways to alter the load order.

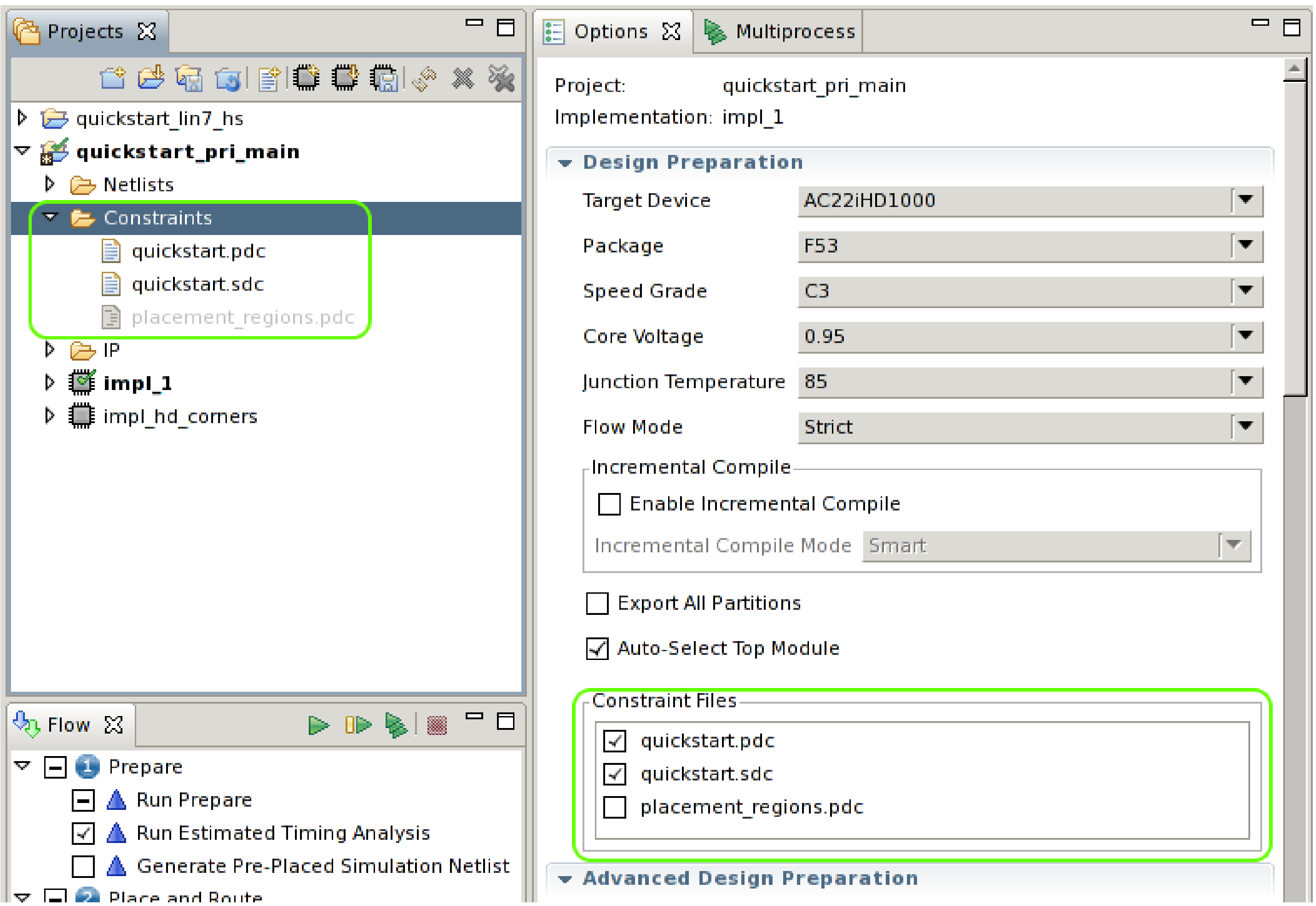
- Changing the order of existing netlist source files and constraint source files can be done quickly using mouse drag-and-drop operations in the Projects View (see page 238) (or by using Tcl commands created explicitly for this purpose). Users should expand the tree to show all the netlist and/or constraint 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 user Runs the Flow (see page 361), the constraint files will be loaded in the chosen order. See also: get_project_netlist_files (see page 564), move_project_netlists (see page 570), get_project_constraint_files (see page 563), move_project_constraints (see page 570).
- 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 357), remove_project_netlist (see page 572),

remove_project_constraints (see page 571), remove_project_ip (see page 572)) and adding them to the project again, one at a time, in the desired order (see add_project_netlist (see page 536), add_project_constraints (see page 535), add_project_ip (see page 535)).

Enabling/Disabling Constraint Files for Implementations

Implementations (see page 312) are allowed to individually enable and disable the loading of constraint files within their owning Project (see page 312). This selective loading is managed through the Options View (see page 215), 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 316) are displayed in grey (instead of black) within the Projects View (see page 238).



Removing Source Files

To remove a source file from a Project (see page 312) in the workspace:

1. Select a source file in the Projects View (see page 238).
2. Click on 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 user's responsibility to clean up unwanted files on disk. The source file is left on disk so that it may be loaded again later if desired.

See also: remove_project_netlist (see page 572), remove_project_constraints (see page 571), remove_project_ip (see page 572), Adding Source Files (see page 355)

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 312) within a project.

1. Select/activate an implementation within the Projects View. The Options View (see page 215) will be updated to show the implementation options for that implementation. At the bottom of the **Design Preparation** implementation options category, there's 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's a list displayed of the Constraint Files in the project.
3. Deselect (uncheck) 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 on 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.

Creating Implementations

To create a new implementation (see page 312) in a project (see page 312) in the workspace:

1. Select a project in the Projects view (see page 238) .
2. Click on the **Create Implementation** toolbar button ( ) in the Projects view.
3. In the Create Implementation dialog (see page 270) , type in the name of the new implementation and click **Finish** .

After clicking **Finish**, the new implementation now appears under the selected project in the Projects view. The new implementation is set to be the active implementation (see page 316) and contains default values for all implementation options (see page). 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 on the **Save Implementation** toolbar button ( ) in the Projects view.
5. In the Save Implementation Dialog (see page 280), type in the file path to the Acxdb Archive File to save the implementation's data to and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation will be stored in the Acxdb Archive file, which can be restored again later.

See also: Restoring Implementations (see page 359), `save_impl`, `restore_impl`

> **Some Flow Steps Automatically Save the Implementation State**
>
> A subset of the Flow Steps (see page 317) will automatically save the current state in Acxdb files. These files will be called `<implementation_name>_prepared.acxdb`, `<implementation_name>_placed.acxdb`, and `<implementation_name>_routed.acxdb` (created at the end of the Run Prepare, Run Place, and Run Route flow steps, respectively).

Legacy Support for Acx Files

As of ACE 3.0, Place and Route data is no longer saved and restored using `.acx` files as in previous releases. Loading and saving place and route data is now done using the new save/restore implementation functionality with Acxdb Archive Files.

Support for the old `.acx` files has been removed from the ACE GUI actions, but can still be accessed via the TCL command interface. If you would like save place and route data to an `.acx` file, you can use the `save_place_and_route` command.

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 on the **Restore Implementation** toolbar button ( ) in the Projects view.
3. In the Restore Implementation Dialog (see page 279), type in the file path to the Acxdb Archive File to restore the implementation's data from and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation will be 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 579), save_impl (see page 589)

Legacy Support for Acx Files

As of ACE 3.0, Place and Route data is no longer saved and restored using `.acx` files (by default) as in previous releases. Loading and saving place and route data is now done using the new save/restore implementation functionality (described above) using Acxdb Archive Files.

Support for the old `.acx` files has been removed from the ACE GUI actions, but can still be accessed via the TCL command interface. If you would like load place and route data from an Acx file, you can use the load_place_and_route command.

Recommendation

After restoring an implementation from a legacy `.acx` file, save the implementation to an Acxdb Archive File to migrate to the new save/restore process.

Copying Implementations

To create a new implementation (see page 312) that is a copy of an existing implementation,

1. In the Projects View (see page 238) , select (activate (see page 316)) 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 270) ,
    a. type in the name of the new implementation
    b. check the **Copy Option Values from Active Implementation** checkbox
    c. select the **Finish** button

After clicking **Finish** , the new implementation now appears under the selected project in the Projects view. The new implementation is set to be the active implementation (see page 316) and contains implementation options (see page) 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 316) in the GUI , do one of the following:

- Single-click on an implementation in the Projects view, activating the selected implementation.
- Single-click on a project in the Projects view, activating the selected project's first implementation.

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 on 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 once the project is saved.

Configuring Implementation Options

To configure implementation (see page 312) options in the workspace:

1. Select an implementation in the Projects view (see page 238), changing the active implementation to the selection.

2. In the Options view (see page 215), use the controls to configure the available implementation options for the active implementation.

After changing implementation options in the Options view (see page 215), the flow status (see page 321) is cleared. A change to an implementation option requires the flow (see page 316) 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 on the output file in the Projects view. The output file now 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 on 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.*

Running the Flow

A flow can only be run on the current Active Implementation (see page 316). If no active implementation is set in the Projects View (see page 238), then the Flow Steps (see page 317) in the Flow View (see page 187) 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 un-checking the checkbox to the left of each flow step label.

Running the Entire Flow

To run the current Active Project and Implementation (see page 316) through the entire flow (sequentially run each of the Flow Steps (see page 317) 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 187), 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.

Disabled flow steps are skipped (not executed) during this operation.

As each individual flow step is run, its Flow Status (see page 321) 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` (see page 580), `enable_flow_step` (see page 550), `disable_flow_step` (see page 544)

Special note regarding the Flow and Incremental Compilation

When Incremental Compilation is enabled, it may sometimes be necessary to recompile all the partitions. This can be done by managing the individual partitions using the Partitions View (see page 231), but an easier way to trigger this is to select the Flow View's context menu choice **Re-Run Flow with "-ic init"**. This will re-initialize the state of all partitions before starting the full flow.

See Using Incremental Compilation (Partitions) (see page 443) for more details. See also: `run -ic init` (see page 580)

Special note regarding Evaluation Flow Mode

When the **Flow Mode** implementation option is set to the value **Evaluation**, the flow steps under **Design Completion** and **FPGA Programming** will not be executed. See Flow Mode (see page 321) for more details.

Running a Sub-Flow

When using the Flow View (see page 187), there are several ways to run a subset of the available Flow Steps (see page 317) on the Active Project and Implementation (see page 316).

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 321) (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, a **double-click** on the chosen flow step will do the same thing.

If any prerequisite required flow steps have not yet been executed, they will be 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, then the chosen flow step will be executed.

Note that this will run the selected step even if that step is optional and not currently enabled (its checkbox is unchecked).

Prior optional flow steps are ignored!

Note that the **Run Selected Flow Step** action executes not only the selected step, but also any preceding required steps. Again, only the preceding ***required*** flow steps will be run, not any preceding optional steps, even if they've been selected (had their checkboxes checked).

See also: `run -step <id>` (see page 580)

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 353) from a saved `.acxdb` file, ACE can continue the flow if the **Resume Flow** ( ) action is chosen. This will cause 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's toolbar, or from the right-click context menu.

If the current flow mode is set to **Evaluation**, the flow will stop after the **Place and Route** category completes. See Flow Mode (see page 321) for details.

See also: `run -resume` (see page 580), `enable_flow_step` (see page 550), `disable_flow_step` (see page 544)

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's **Stop Flow** ( ) action.

Some flow steps may respond by stopping immediately, while others will need to perform some additional work before exiting the flow step. In both cases, the Flow Status (see page 321) of that step will typically be 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 will show many logged error messages for the interrupted flow step. Typically the user will then be able to **Resume Flow** ( ) or **Run Selected Flow Step** and ACE will resume normal work from the last successfully completed flow step.

Running Multiple Flows in Parallel

Normally, ACE only allows a single project (see page 312)'s implementation (see page 312) to be run through the flow (see page 316) at a time. Using the Multiprocess View (see page 204), ACE allows users to run 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 323) of the resulting frequencies, permitting the user to make QOR performance comparisons between implementations utilizing different starting clock constraints, placement constraints, and potential optimizations.

Finding the Multiprocess View

To make use of the Multiprocess View, (which is hidden by default,) the view must first be made visible. To show the Multiprocess View, select the Projects Perspective (see page 23) ( ). Then, in the Flow View (see page 187), select the **Show Multiprocess View** ( ) button. This will cause the Multiprocess view to be displayed, and will also hide /minimize 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 will again become 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)" section allows the user to configure 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** will cause 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 302).

License Management Considerations with Multiprocess

Warning

Each parallel ACE process needs its own ACE software license. When running using the Multiprocess View in the ACE GUI, if the user wishes to run *N* parallel execution queues, then the user will need *N*+1 ACE licenses (the extra license is for the GUI itself, as it is managing all the queues running in the background). Users should talk to their Achronix FAE to ensure their site has enough licenses to enable running with Multiprocess functionality.

The following is a common best practice when determining the needed ACE license counts to support multiprocess runs at customer sites, as well as choosing the best value of **Parallel Job Count** based upon the available 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.

At customer sites trying to minimize their license usage, or where users must share the available execution hosts equally, the minimum number of ACE licenses (L_{min}) required would then be $L_{min} = U + P$. Each user is then allowed to consume up to L_{user} licenses during their multiprocess sessions, where $L_{user} = 1 + (P \div U)$.

At customer sites wanting to maximize job throughput, where individual users may be allowed to completely saturate the execution hosts, the maximum number of ACE licenses (L_{max}) required 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 will 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 will actually take 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 don't exhaust the physical memory (RAM) available on the executing workstation, otherwise flow execution times will quickly increase (due to the OS swapping memory pages to disk). Don't forget to 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 will take multiple Gigabytes (GB) of memory - the exact amount will vary depending upon the size of the design and the size of the target Achronix device. (Smaller designs and smaller devices will, of course, take less memory.) A guesstimate for large designs on a very large FPGA device is around 16GB of memory used for each background process. Again, this is a guesstimate – designs nearing 100% device utilization may require more memory.

Be aware that with modern multi-core hyper-threading workstations, memory limits are usually going to be 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 guesstimate 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 were 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 worth the user's time to track the total multiprocess duration at multiple parallel process counts, so as they continue working in the future, 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* * (*C* * *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's 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's design requiring a peak memory usage of 7GB, then the memory constraint value is (27.5GB / 7GB =) about 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 317) 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 Tcl command `get_ace_peak_memory_usage` . These features should allow the user to make an educated decision as to how much memory each parallel background process will require for their 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's 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.

- A 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 will be necessary for the user to write their own script or executable which approximates synchronous/blocking functionality. In theory, this should be possible by submitting the job, capturing the unique identifier for that job, looping while querying the job status (using the previously captured job's unique identifier) from the job system until completion is indicated, capturing the job's exit code, and then returning the appropriate exit code (to ACE) indicating the job's success/failure 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's job submission process finishes before the submitted ACE job is complete (as would happen with a non-blocking job submission system), the ACE implementation's output files will be either missing or incomplete when queried, and the Multiprocess Summary Report will show 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') to the (blocking) job submission process. It is expected that this will also kill/cancel 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 on their job submission tool.

Job Working Directory

In some cases it may 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 will change for each implementation, and will typically be 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 variables in italics replaced by their logical values:

`ACE_Multiprocess_username_projectname_implementationname`

Additionally, special characters found in the variable's values will be 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're running.

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's 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 302), 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 through the `qsub` command. (When using a system other than the GridEngine, users will 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's execution hosts.

ACE is able to optionally provide some values to the job submission system if the related argument fields are populated. These 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 will be required by the job submission executable for it to meet ACE's 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 will fail.

Debugging Job Submission System Configurations:

If the job submission system is properly configured on the host machine, (meaning the user is able 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

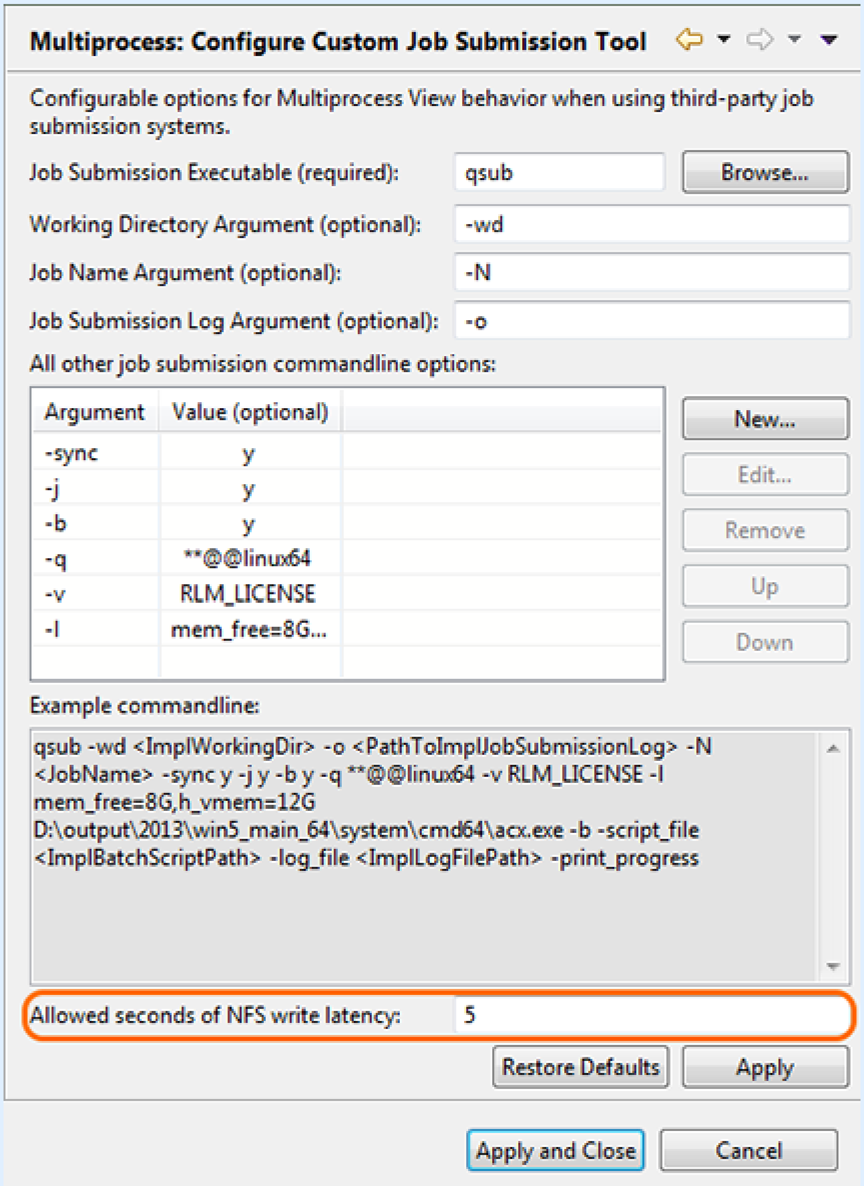
Attempting to manually run the logged command on the command-line (without the Multiprocess View's additional automated safety locks in place) may cause ACE datafile corruption.

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's 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 323) will show the message "No Timing Results Found" for that implementation.

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 via the Multiprocess user preference **Allowed seconds of NFS write latency**, circled in the screenshot below.



Configuring the Desired Flow to be Followed by the Selected Implementations

All the implementations run through the Multiprocess View will follow the same flow steps (see page 317) through the flow (see page 316), as configured in the Flow View (see page 187). Thus, users must ensure 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)", users may choose 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, users may choose 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 will be 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 will be 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 will 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

In the Projects View (see page 238), select the desired project (see page 312). The Implementation Table within the Multiprocess view's "Select Implementations (see page)" section will be updated to display data for the active project and implementation (see page 316).

In the Multiprocess View, ensure the **Existing Implementations** radio button within the "Select Implementations" section is selected. This will limit 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) is covered in Attempting Likely Optimizations Using Option Sets (see page 434).)

In the Implementation Table, all listed implementations will be 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 isn't 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 will cause the table to resize to exactly fit the current implementation list.

Starting Background Execution

Once the parallel count has been set, the flow has been configured, and the desired implementations have been selected, press the **Start Selected** button (or the equivalent **Start Background Queue Execution** ( ) action in the Multiprocess view's local button-bar or menu) to begin background multiprocess execution.

After multiprocess execution has been started, the **Parallel Queue Count** and Implementation Table will be disabled. They will not be re-enabled until multiprocess execution is completed. In the "Multiprocess Run Logs (see page)" section, a new tab is created for each selected implementation's logged output. 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 315) for each implementation.)

As implementations are queued, start execution, and complete execution, the implementations' execution states (see page) will be updated in the implementation table, and each implementation's log tab icon will also be 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 will be queued and allowed to run once multiprocess execution is completed. Similarly, multiprocess execution will be blocked until all in-process and already-queued ACE Tcl commands (including running the Flow in the foreground) are completed.

Stopping/Canceling Background Execution

Users may quickly cancel all queued and executing background implementations by selecting the **Stop All** button below the Implementation Table, or the equivalent **Stop All Background Queue Execution** ( ) action in the Multiprocess view's 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 will display 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 will cancel/stop execution of that task. Because the Progress View can list more tasks than just the background multiprocess implementations, caution should be used when canceling tasks; users do not want to cancel/stop the wrong task.

Viewing the Results

After the first implementation completes execution, an HTML Multiprocess Summary Report (see page 323) is created and opened in ACE. (The report file is created in the project directory, and is named `multiprocess_summary.html`. This will automatically overwrite previous multiprocess summary reports without prompting.) As each subsequent implementation completes execution, the multiprocess summary report will be updated with the latest data.

As implementations complete execution, their execution states (see page) change appropriately. If an implementation encounters errors while running the flow, that implementation's execution state becomes the Error state, which will be 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 will be updated to include a summary of the captured error messages. Error details will be visible in the log messages shown in the tab, as well as within the Implementation Log (see page) and Multiprocess Log (see page) for that implementation.

Caution!

There is a known sequence whereby all multiprocess results are identical. If the user has an existing project for which they have already generated option sets, and then the user upgrades to a newer version of ACE, ACE will prompt the user when opening the existing project to reset the implementation (see page 312) 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 will only applied by accepting this reset. At the same time, older implementation options that have been deprecated will be removed.

The issue is that currently ACE will reset all of the option sets (see page) to the same default values. Subsequently when a multiprocess flow is run with the new project, all results will be identical. The workaround is after having upgraded ACE to the new version and accepting the implementation (see page 312) option reset, to then delete all the implementations other than the original base implementation and to then regenerate the option sets (see page).

Multiprocess Batch Mode

Overview

In order to obtain the highest QoR, ACE supports the ability to run multiple different implementations in parallel using Multiprocess. Multiprocess is available from the ACE.GUI and is described in Multiprocess View (see page 204) (see also Running Multiple Flows in Parallel (see page 363) and Attempting Likely Optimizations Using Option Sets (see page 434)). Running Multiprocess from the GUI supports different implementations, allowing varying options can be tested against the design. Furthermore, Multiprocess can be targeted to a user's grid engine to enable the multiple implementations to run in parallel.

Multiprocess batch mode provides the ability to run Multiprocess from the ACE Tcl console command line or by using an external Tcl script. Multiprocess batch mode supports both the varying implementations the same as from the GUI, but also adds the ability to do a seed sweep, varying the starting seed for a particular implementation. Multiprocess batch mode also supports grid engine submission in the same manner as the GUI.

Multiprocess batch mode is called within the ACE Tcl console or within the ACE batch shell using the command run_multiprocess (see page 583).

> **Note**
>
> In order to run Multiprocess batch mode from a terminal or shell, the directory containing the ACE executable must be included in the path.

Modes

Similar to running Multiprocess from the GUI, Mutliprocess batch mode has to be run in the context of a currently Active Project and Implementation (see page 316). 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;

- Generate implementations from option sets (default setting)
- Seed sweep (`-seed_sweep`)
- Use existing implementations (`-use_existing_impls`)

A full list of all the options is given in the run_multiprocess (see page 583) 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 will be overwritten. See Attempting Likely Optimizations Using Option Sets (see page 434) for additional details.

The currently active implementation is always included as one of the executed flows when this mode is used.

> **Warning!**
>
> When generating implementations from options sets, users are recommended to set the current implementation to be the base default implementation of the project (by default named `impl_1`). The options sets have been created to give an optimum range of strategies to achieve best QoR when based on a default implementation. If the user uses an already modified implementation, possibly from an earlier multiprocess run, as the base, then the option sets will be applied to this modified implementation. This recursive use will bias the effectiveness of the option sets, leading to a reduced variance in the strategies, and will probably lead to sub-optimal QoR.

Seed Sweep

> **Seed Sweeps are only available in batch mode**
>
> The seed sweep automation is not yet available within the GUI's Multiprocess View (see page 204). Seed sweeps are presently only available in Multiprocess batch mode.

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 will be named `{active_impl_name}_seed#` , e.g., `impl_1_seed18` for a seed value of 18 and an active implementation name of `impl_1`.

The currently active implementation will always be 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 will simply run 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 implementation, specify:

```
run_multiprocess -use_existing_impls [get_project_impls]
```

Unlike the other modes, the currently active implementation *will not* be 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 `-stop_flow_at`. This argument can be used terminate each flow at a particular step; for example if `report_timing_routed` is specified, then none of the DRC or bitstream flow steps will be performed. Using this argument reduces the overall time taken for Multiprocess batch mode, as each implementation will run 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 will be explicitly enabled.

Getting Started

The commands to start Multiprocess batch mode vary according to whether the user is running ACE 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 344)
2. Use restore_project (see page 580) to load the project.
3. Set the active implementation (see page 592)
4. (Optional). Use disable_flow_step (see page 544) 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 583) command. See examples (see page 375) 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 344)

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, the user must use CTRL+C.

ACE GUI

1. Open ACE GUI
2. Load the project (see page 353)
3. Set the active implementation (see page 360)
4. (Optional) Using the Flow View (see page 187) select or deselect any flow steps desired/needed or bypassed for all of the implementations that are to be run.
5. In Tcl console window, issue run_multiprocess (see page 583) command. See examples (see page 375) 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

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, users can use the Multiprocess Summary Report (see page 323), `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 an command shell, or from within the ACE Tcl console. This report can either be viewed external to ACE using a web browser or else within ACE as detailed below.

Viewing Multiprocess Summary Report within ACE

Open the Multiprocess Summary Report from the Projects view, by right-clicking on the project.

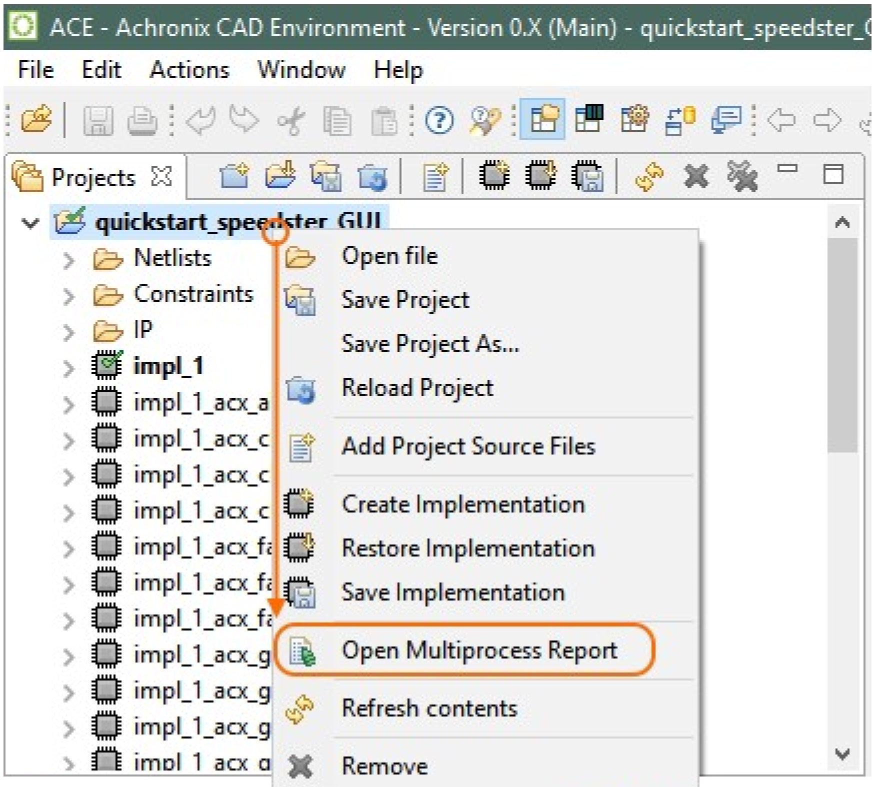


Figure 42: *Open Multiprocess Report*

The report can be refreshed by right-clicking in the view and selecting 'Refresh', or by clicking the refresh hotkey, **F5,** when the report tab has focus (click on the report first). The report view will not automatically update when using Multiprocess batch mode. This behavior differs from when Multiprocess is run directly from the GUI, when the report view is automatically updated.

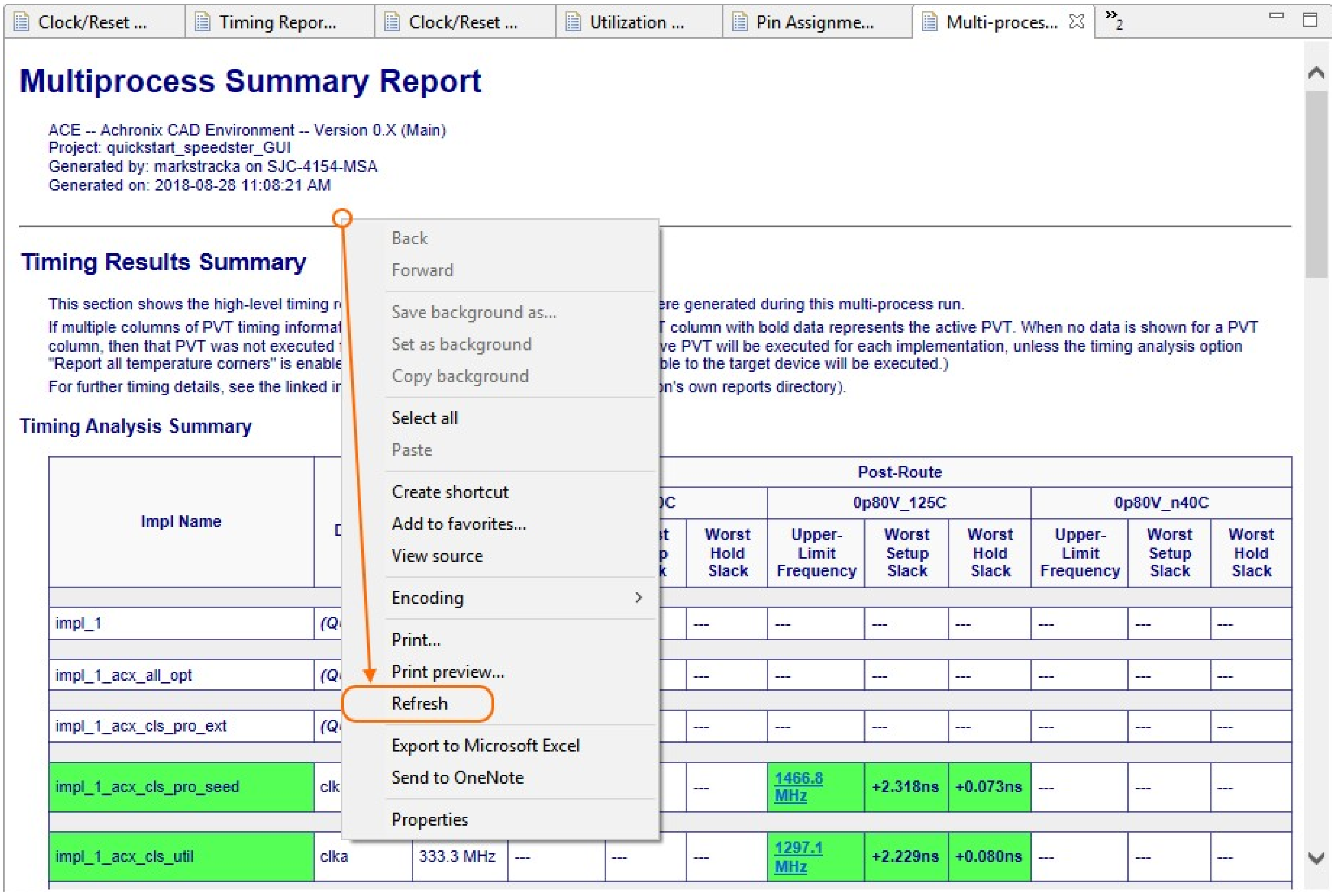


Figure 43: *Refresh Multiprocess Report View*

Furthermore, progress monitoring can be achieved if a grid engine is used — the user may be able to monitor completed jobs from the grid engine. Finally to see the status of an individual implementation, the user can open the `<implementation>/log/impl.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 window to cancel the process.

Detecting Changes to Project Source Files

ACE provides a rich set of features to enable users to detect changes to their 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 will appear and ask if the user wants 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 do the checking, we cache 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 user copies a project 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 you save the state of the ACE database 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

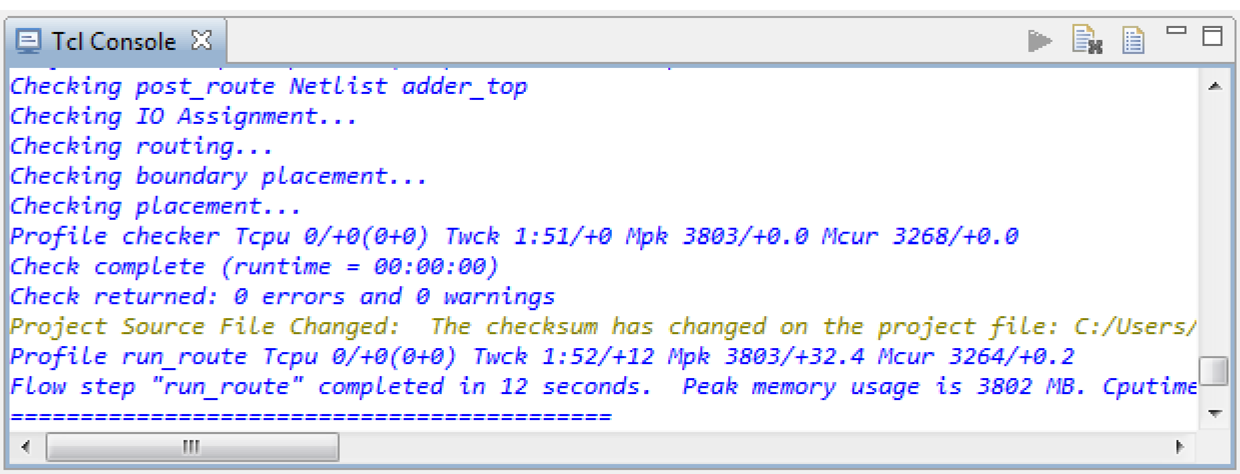
When ACE loads/restores an ACXDB file (when you load saved place and route data from the ACXDB file on disk into the ACE database for the active implementation), ACE checks all project source files and checksums on disk against the cached project source files and checksums inside the ACXDB file. If a 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 will print 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 in to the ACE database from disk. Whenever the Run Prepare flow step is run, ACE will cache the project source files and checksums for all files used in the active ACE project implementation.

Automatic Checking while Running the Flow

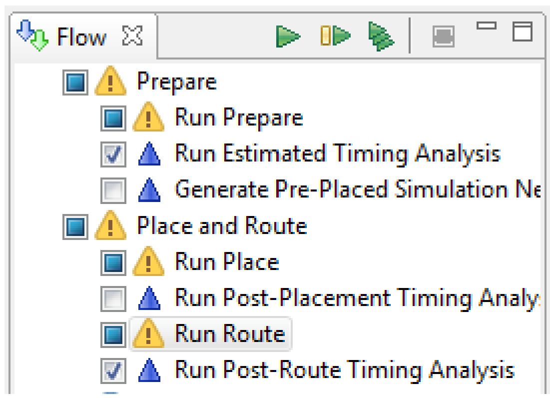
Each flow step (Run Place, Run Route, Timing Analysis, Final DRC checks, etc) will check the project source files (including the ACXPRJ ACE project file itself) and checksums on disk against the files and checksums cached at the beginning of Run Prepare. If any file is missing, added, or out of sync, ACE will report a warning at the end of each flow step to the Tcl Console and ACE log file.



Warning Visualization in the Flow View

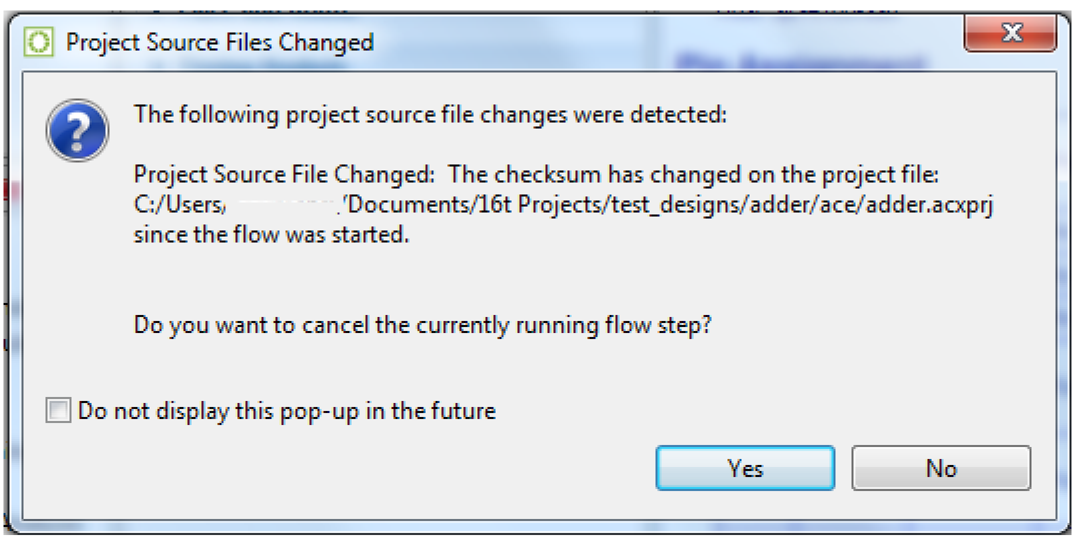
If any flow step reports a warning about out-of-sync files, all completed flow steps in the Flow View (see page 187) will be marked with a yellow warning icon instead of the green checkmark icon to indicate that the step is complete, but is out of sync with the source files on disk. The tooltip text in the Flow View will show all the warning messages.

Even when no flow step Tcl commands are running, the GUI will check 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 will be 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.

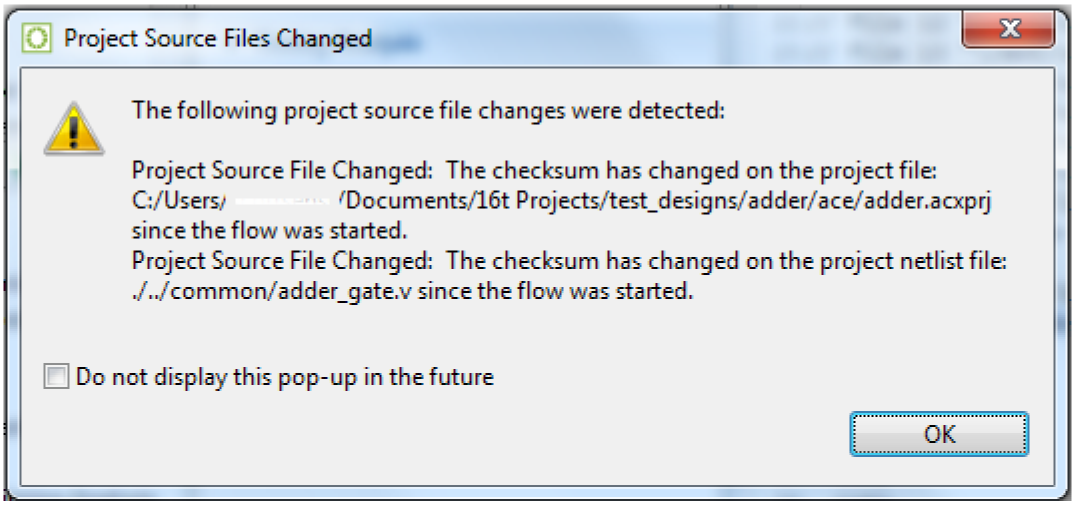


Pop-up Dialog Warnings

If the flow is running, a “Project Source Files Changed” dialog will pop up 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 will be displayed in the dialog. The user can choose to cancel running the rest of the flow, or to let the flow continue. The flow continues to run in the background until the user makes their choice. The pop-up dialog has a checkbox for a user preference which allows the user to disable the pop-up from being shown in the future.



If the flow is not running, a different “Project Source Files Changed” dialog will pop up 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 will be 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 checkbox for a user preference which allows the user to disable the pop-up from being shown in the future.

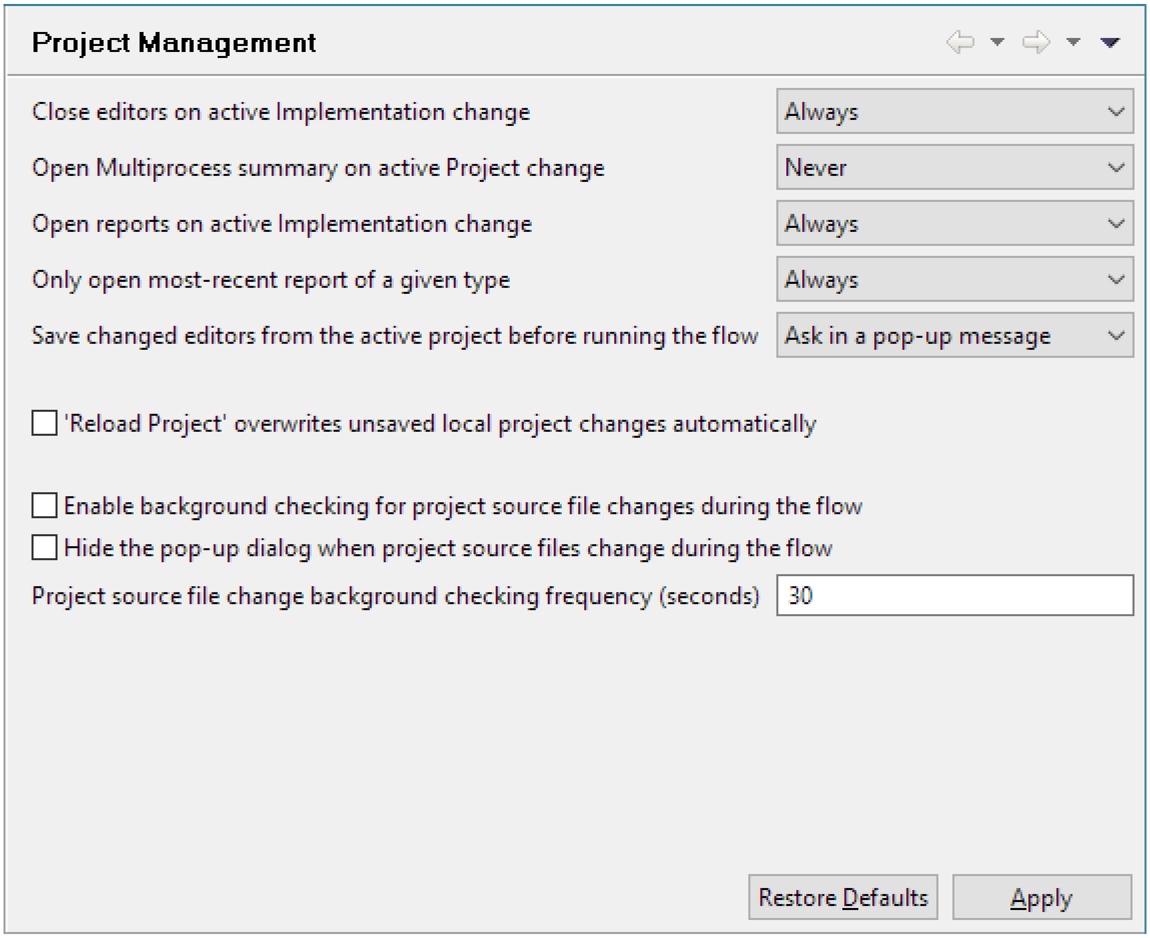


Minimal Pop-up Interruptions

In general, pop-ups are minimized to only alert the user of new changes. The Project Source Files Changed dialog will pop up only when a new change is detected. So if you change the gate level netlist file while the flow is running, you will see the pop-up (if enabled by the user preference). If you then change the same gate level netlist file several more times, no further pop-up will appear since the user has already been notified that the file is different than the original source file. However, if you then change a project constraints file (in addition to the gate level netlist), the Project Source Files Changed dialog will pop up again to alert the user 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 306). This is the same preference that is controlled with a checkbox in the Project Source File Changed pop-up. 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 user chooses to disable it with the checkbox in the dialog.



Tcl Command Support

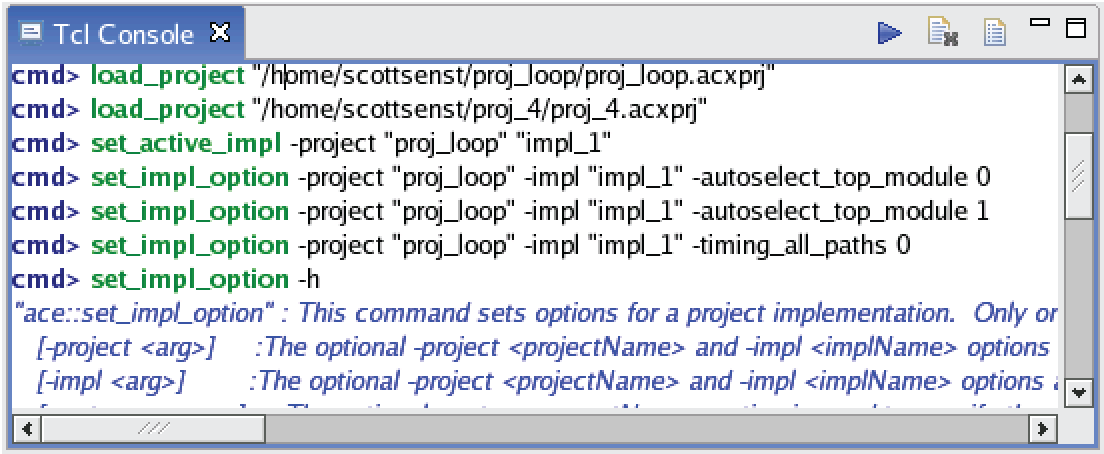
The `check_project_status` Tcl command that can be called to manually check file and checksum consistency outside of the built-in checks done at the end of each flow step. If any file is missing, added, or out of sync, ACE will report a warning at the to the Tcl Console and ACE log file. This command only applies if the user 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 258) 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 258) to do the work. All Tcl commands generated by GUI actions are displayed in the Tcl console along with any output from the command.



ug001_c08_08_012108

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 on 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.

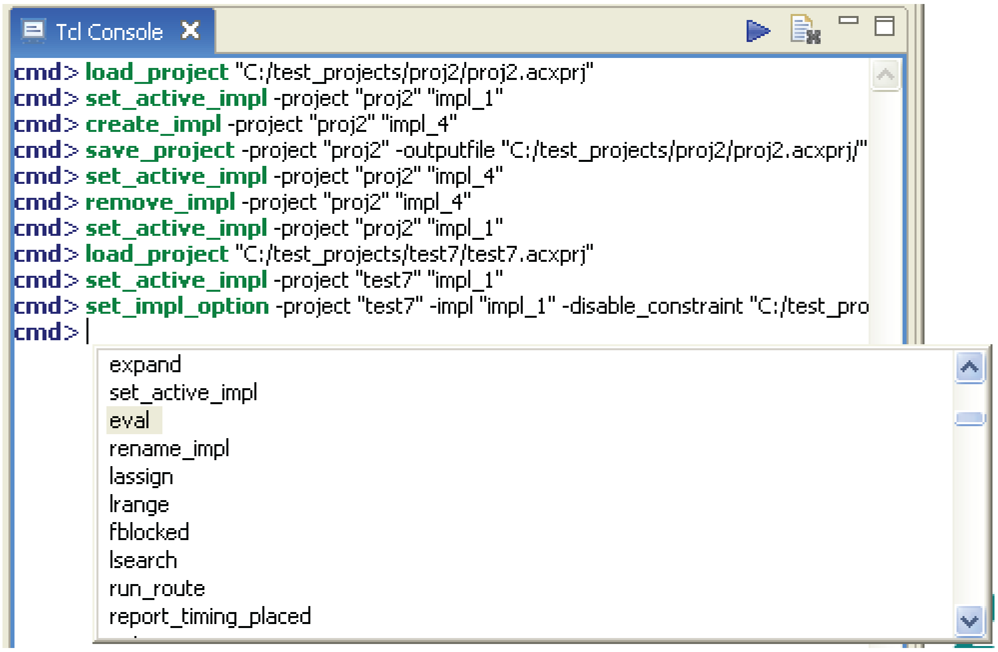
Command Highlighting

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 in the TCL console, pressing **TAB** 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 will be heard.

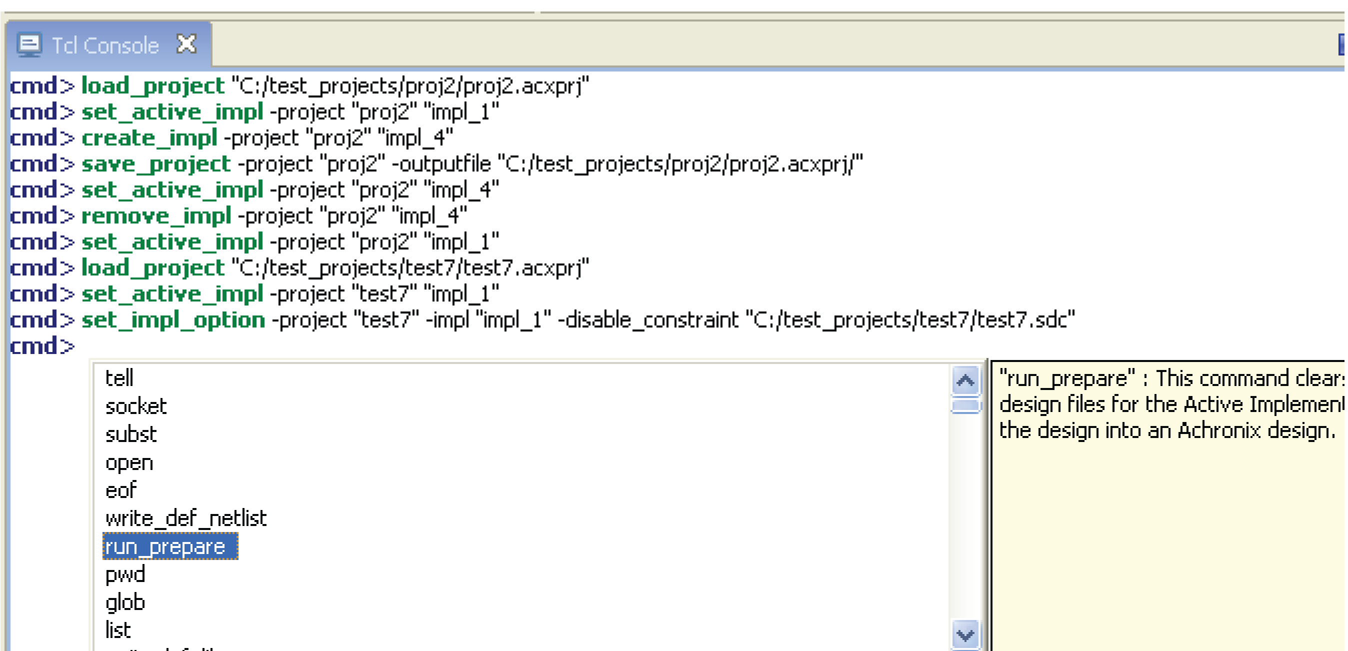
Pressing **TAB** at an empty **cmd>** prompt pops up the full list of available commands. When the command auto-completion dialog is open, use the arrow keys to navigate up and down the list of choices and press **ENTER** on a selected command to complete it at the command prompt. Typing while the command auto-completion dialog is open shortens or lengthens the list of valid commands, depending on the cursor position in the TCL Console view.



ug001_c08_04_v01

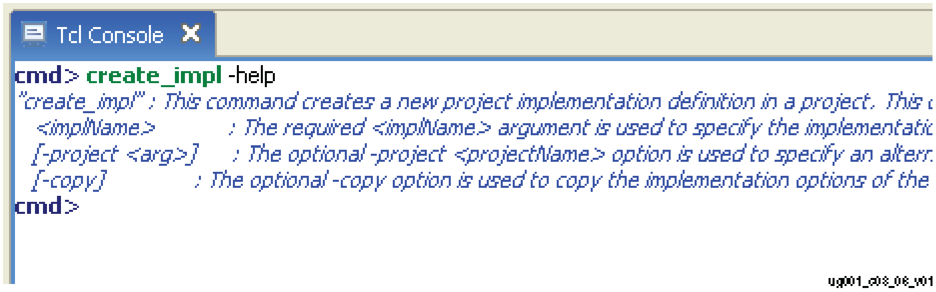
Command Help

When the command auto-completion dialog is open, help text appears to the right of the command list for the selected command.



ug001_c08_05_v01

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** option to output the help text to the TCL console.



Text Limit

The TCL Console view has a limit of 2000 lines. Once 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 will be visually truncated at 500 characters in the console. The actual returned value will not be edited, just the textual representation shown in the console.

Performance Tip:

Performance of the TCL console is much higher when there are fewer than 2,000 lines of text.

Hitting the text limit will not clear the contents of the ACE log file. All messages will continue to be logged in the log file, and earlier messages will not be removed.

Clearing the Console

Text in the TCL Console view can be cleared by clicking on 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.

Performance Tip

Clearing the console increases console messaging performance.

Clearing the console will 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 reach 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 on 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 between, for example, a net and a pin with the same name.

Many Tcl commands (like `select`) require that these prefixes be used when commands are issued. Other commands (like `find`) will, by default, include these prefixes on the return values.

Table 165: ***Object Type Prefixes Used in ACE Tcl Commands (Sorted Alphabetically by Prefix)***

Prefix	Object Type
`b:`	Package Ball
`c:`	Critical Path
`d:`	Device Port
`f:`	Fabric Pin
`i:`	Instance
`k:`	Clock Domain
`n:`	Net
`p:`	Port
`s:`	Site
`t:`	Pin

Table 166: ***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:`
Package Ball	`b:`

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 25) and Views (see page 163) which work together to guide the user 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 will generate 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's design for synthesis and simulation.

> Note that 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 23) ( ). In addition to the IP Configuration editors, this perspective incorporates supporting views allowing the user to:

- Create new IP configurations (IP Libraries View (see page 197))
- View a graphical diagram of the IP Configuration currently being edited (IP Diagram View (see page 196)); the diagram may show the macro's interface, the dataflow, and/or the placement of the IP instance within the chip.
- Navigate instantly to any page of the active IP Configuration Editor, while displaying the names and validity of each page (Outline View (see page 225))
- View a detailed list of all the errors and warnings pertaining to all IP Configuration files currently opened (IP Problems View (see page 198)). This view also allows the user to navigate directly to the source of the problem in the relevant IP Configuration Editor.

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 197)) to open the New IP Configuration Dialog (see page 277). 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 will be used as the module name for the generated Verilog module. The user must choose a name 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` , then the generated Verilog module will be named `module foo`. If the user names the `.acxip` configuration file `LRAM.acxip`, then the generated Verilog module will be named `module LRAM.` If `LRAM` is a primitive in the Achronix libraries, then the user design will error 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 225) 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 198). Some IP Editors will also display supplemental graphical information in the IP Diagram view (see page 196).

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 that field's value. The green checkmark ( ) indicates the value in the field has no problems. A warning ( ) or error ( ) icon is shown when the field's value does have one or more problems. The tooltip of 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 may 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 fields' 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

Users should make modifications to fields within the IP Configurations Editors in a top-down, left-to-right order. When users edit 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's background color will change to indicate a problem. Just like when user-editable IP configuration properties fall outside a legal range, an IP Problem entry (see IP Problems View (see page 198)) is created. But an IP Problem created by a calculated value field will not "blame" the calculated value field, it will instead blame 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.

For example, when one of the PLL's four "clkout Output Frequency" calculated values is outside its legal range, it blames the user-editable "Refclk Input Frequency (MHz)". While the "Refclk Input Frequency (MHz)" is just one of the user-editable fields involved in the calculation (there are over a dozen other user-editable fields potentially involved), it is always relevant, and the cyclic nature of the circuit, particularly when in external feedback mode, makes it impossible to pick a single most-relevant user-editable field.

> **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

The user may 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 225). The user may 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 the user's changes to the IP configuration's editable fields.

The user may left-click on any text in the IP Diagram View (see page 196) to turn to the IP Configuration Editor page containing the settings for that text.

The user may double-click a table entry in the IP Problems View (see page 198) 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 274). 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.

> The generated VHDL RTL wrapper is not standalone. It requires the generated Verilog RTL file.

Adding Configuration Files to a Project

If the option **Add to active project** in the New IP Configuration dialog (see page 277) box was not selected, the configuration files must be manually added to the active project. Use the procedure under Adding Source Files (see page 355) to add the configuration file and its related source files to the active project.

Live Link Tuning for SerDes and Derived Interfaces

Live Link Tuning using the ACE GUI is presently only available for Speedster FPGA devices

Live Link Tuning is not available for other Achronix product types.

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.

Note that other reference documents will be available for alternate (non-GUI) link tuning procedures, including (in some cases) automated tuning that may be managed from within the logic of user designs (without requiring manual user intervention).

The remainder of this section of UG001 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 may 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 will 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 the user to compute 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 23) and open the existing ACXIP file for the SerDes, Interlaken, or Ethernet interface. This will open the appropriate IP Configuration Editor for the chosen IP. While each of these editors includes its own Link Tuning Page (see page 146), 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.

A properly configured ACXIP file is required before beginning Link Tuning

It is assumed that customers will use 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, users must have an existing ACXIP file before commencing GUI-managed Link Tuning. If users don't have an ACXIP file, they can create a new SerDes, Interlaken, or Ethernet ACXIP file (see Creating an IP Configuration (see page 387)) and configure it 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's Link Tuning process.

The GUI's Link Tuning pages interact with a live FPGA through the Bitporter JTAG interface. It will be 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 432), a necessary precursor to Link Tuning.

> **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 288). See Configuring the JTAG Connection (see page 411) 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's pages must be properly configured for the interface which will 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 387)
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 388).
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 389).
4. Run the design through Synthesis, and then Place and Route. When timing is met, generate a bitstream. See Running the Flow. (see page 361)
5. Make sure all ref clocks are connected and running on the board, including for each of the SerDes lanes used in the design.

   > **Don't forget the ref clocks!**
   >
   > Connecting and running the ref clocks to the utilized SerDes lanes is the detail most users forget 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 196) which shows the exact lanes being used by the current ACXIP file. As an example, see the diagram for the Speedster22i SerDes Configuration Editor (see page 120).)

6. Program the FPGA with the generated bitstream (from step 4). See Playing a STAPL File (Programming a Device) (see page 432).
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 will be 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.

> **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 user must first fix the issue with the PMA status before continuing. The fix may involve manually changing some Tx Driver or Rx Eq settings to get the status to become green (good).

10. Once the PMA status is good, observe the initial FOM for each lane. Cautious users may want to write down the FOM values, along with the used Rx and Tx settings used. Taking screenshots is an easy alternative. (ACE presently does not log the user's full tuning process.)
11. Press **Run Rx Auto Eq** and wait for it to complete.

> **Users should not run Rx Auto Eq tuning 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 will result in bit errors. Rx Auto Eq tuning should only be done 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. Cautious users may again wish 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

Opening and Closing the Floorplanner's Fly-Out Palette

To open and close the Floorplanner view's fly-out palette of view options:

1. Click on the **Fly-out** button (◀) on the far right side of the Floorplanner View (see page 178) 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. Once the view options are configured, click on 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 178).

> Zoom levels are always in powers of 2, i.e. zoom in is at 200% and zoom out is at 50%. Therefore, it may 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 over the point for zoom in or out from in the Floorplanner view.
2. Slide the mouse wheel forward to zoom in or slide the mouse wheel backward to zoom out.

To zoom in and out with key-strokes:

1. Hover the mouse over the center of the area for zoom in or out from 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 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 to zoom out from 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 over the point to zoom in or out from in the Floorplanner view. Single-click the left mouse button to zoom in or single-click 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 to zoom in to or out from in the Floorplanner view.
2. Press the **Zoom In** button ( ) to zoom in or the **Zoom Out** button ( ) to zoom out.

Floorplanner Panning

To pan with the scroll bars:

- 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.
- In Linux, place the mouse cursor over a scroll bar, then roll the mouse wheel.

To pan with key-strokes:

- Use the **ARROW** keys on the keyboard to pan left, right, up and down.
- 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 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.

Performance Tip

If panning the view is too slow, in the Floorplanner View Optimizations Preference Page (see page 296) 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 key-strokes:

1. In the "Selection" ( ) section of the view's 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 position to set the current selection (clearing/replacing the previous selection).
    a. Optionally, press and hold the '**SHIFT+S**' keys to start a selection rectangle at the current mouse position to add 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. Select the **Selection Tool** ( ) from the view toolbar.
2. From the "Selection" section of the view's 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 left mouse button on the desired object, or click and drag with the left mouse button in the view to create a selection area rectangle.
    a. Optionally, hold **CTRL** down when clicking/dragging to add to the previous selection instead of replacing the previous selection.
5. Release the mouse button to apply the selection.

Additionally, individual objects may be right-clicked with the mouse, and **Add to Selection** may be chosen from the context menu popup.

Further details about the ACE Selection Set are available at the Selection View (see page 249) page.

Deselecting Floorplanner Objects

To deselect objects with key strokes:

1. Select the **Selection Tool** ( ) from the view toolbar. From the **Selection** section fly-out palette, check the object types to deselect.
2. Press and hold the '**d**' key on the keyboard to start a selection rectangle at the current mouse position to deselect the objects.
3. Drag the mouse while holding down the key to create a selection rectangle including the objects to deselect. Then, 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. From the **Selection** section of the fly-out palette, check the object types to deselect. Also, ensure the Action control is set to **Deselect**.
2. Click and drag with the left mouse button in the view to create a selection rectangle. Then, 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 on the desired mouse tool on the view toolbar.

Filtering the Floorplanner View

It is often useful to filter the Floorplanner View (see page 178) graphics to see only objects of interest.

Filtering with Layers

Simple filtering of the view by object type is done with the **Layer** options in the view's fly-out palette.

By checking or unchecking the individual layers, users may show/hide all members of a given object type. 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 that 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 292) 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 178), a tooltip (hover text) can be enabled. In addition, the contents of the tooltip can be printed to the Tcl Console View (see page 258) for easy copy and paste.

To get object tooltip text:

1. In the **Tool Tip Text** ( ) section of the fly-out palette (see page), enable the checkboxes for the object types whose data should be contained in the tool tip text.
2. In the Floorplanner view, hover over objects to display tool tip text.

Capturing Tooltip Content

Optionally, users may 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 178) (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 that 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 178). Viewing all of the data simultaneously can be overwhelming, so ACE provides tools like Selection and Highlighting so that users may visualize a particular subset of their entire design 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 249). For longer-term visualizations, or to simultaneously compare and contrast multiple design subsets, ACE provides the Tcl `highlight` functionality. 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 like the Netlist Browser View (see page 211), where the highlight colors of instances are displayed in their own table column.

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 385) 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 will next be used from that view. (Each view tracks an active Highlight Color independently of the other views, allowing a user to have 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.

In tabular views (like the Clock Domains View (see page 165)), 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 will be shown for the aggregation's row in the table. The aggregation will only display 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 that 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.

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're 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 292).	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, that view's own ( ) **Choose Highlight Color** will be used.
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 will be saved in `acxdb` files when Implementations (see page 312) are saved. As a result, prior highlights will be restored when an implementation's `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 249)	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's 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 292).)

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 292).) Note that 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 328) section discusses this in further detail.

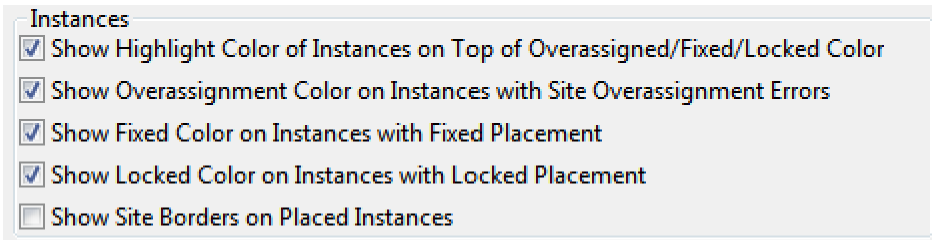


Figure 44: Instance state render management on the Floorplanner preferences page

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 292). 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).

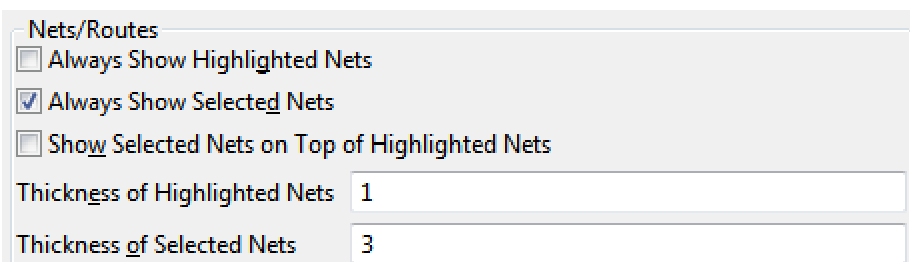


Figure 45: Example (non-default!) Floorplanner preference configuration allowing simultaneous display of both Selection and Highlights for Nets

> **Floorplanner performance warning**
>
> Choosing to render Highlighted Nets thicker than a single pixel wide may have a significant negative performance impact upon Floorplanner rendering speeds of large designs at some customer sites. Exact details will depend upon specifics of the workstation's 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 flag for the highlight (see page 568) 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. Once all batched changes to the highlights have completed, the user must then call `apply_highlights` to send the highlight updates to the GUI for display.

Example Tcl sequence to highlight all instances orange, except instance names starting with 'temp'

```
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

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 pad or macro 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.

> 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 245), Selection View (see page 249), or Netlist Browser View (see page 211):

1. In the **Placement** ( ) section of the Floorplanner View (see page 178)'s or Package View (see page 226)'s fly-out palette, check the placement options desired.

   > Fixed placement is recommended for all pre-placement.

2. Pan and zoom the Floorplanner view or Package 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 or Package view.

   > 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 from within the Floorplanner view or Package view (move from one placement site to another):

> 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 of the destination view, check the placement options desired.

> Fixed placement is recommended for all pre-placement.

2. Click on 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 or Package view until both the to-be-moved instance and its intended destination placement site are visible.
4. In the Floorplanner view or Package 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.

> 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 178)'s fly-out palette, check the object types whose placement 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. Then, 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 178)'s toolbar. From the **Selection** section of the fly-out palette, check the object types whose placement you wish to fix. Also, ensure the Action control is set to **Fix Placement**.
2. 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.
   Note: *Not using* **CTRL** *will clear the previous selection.*
3. Release the mouse button to fix the placement of the activated objects in the 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 178)'s fly-out palette, check the object types whose placement 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 on the keyboard to create a selection rectangle which includes the objects desired. Then, 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 178)'s toolbar. From the **Selection** section of the fly-out palette, check the object types whose placement you wish to fix. Also, ensure the Action control is set to **Un-fix Placement**.
2. 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.
   Note: *Not using* **CTRL** *will clear the previous selection.*
3. Release the mouse button to un-fix the placement of the activated objects in the selection.

Group Placement Mode

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), users only alter the placement of a single instance.

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 will fail.

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's 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), then the coordinate shift is (X=+5000, Y=+10000). For each instance in the selection set, this coordinate shift will be applied to that instance's starting site coordinates; the resulting X and Y values will be the coordinates where the destination site will be 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 will be moved. All instances would be left untouched in their initial placements, including the anchor instance which was dragged-and-dropped.

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's 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 will 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 249)).
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 178)'s fly-out palette) is in the desired state.
5. Enable **Group Placement** mode by clicking the checkbox (within the Floorplanner View's fly-out palette)
6. Choose which placed, selected instance will 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 will have corresponding destination sites.
8. Drag the anchor instance from it's initial placement to the destination site. If all initial requirements were met, and if destination sites were found for all selected instances at the calculated offsets, then 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's fly-out palette)

⚠ **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's **Fixed Placement** checkbox will affect all adjusted placements.

If **Fixed Placement** is checked, then all adjusted placements will now be fixed, regardless of whether they initially had soft or fixed placement. If **Fixed Placement** is unchecked, then all adjusted placements will now be 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 178), Selection View (see page 249), Selecting Floorplanner Objects (see page 394), Deselecting Floorplanner Objects (see page 394), `select`, `deselect`, `set_placement`

Removing Placement

To un-place objects with key-strokes in the Floorplanner View (see page 178) or Package View (see page 226):

1. (If using the Floorplanner view:) Ensure the **Instances** checkbox is checked in the **Selection** ( ) section of the view's Fly-out Palette.
2. With the mouse positioned in the Floorplanner view or Package 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 or Package view:

1. Select the **Selection Tool** ( ) from the view toolbar.
2. In the **Selection** ( ) section of the view's fly-out palette:
   a. Set the Action control to **Remove Placement**.
   b. (If using the Floorplanner view:) 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 245), and Selection View (see page 249).

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 394)), 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 258), 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 399).
2. In the Floorplanner view or Package view, click on the **Save Pre-placement Constraints** toolbar button ( ) to bring up the Save Placement Dialog (see page 281).
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, then the Projects view shows the new files under the active project's Constraints folder.

Using Pre-Placement in the Flow

Types of Pre-Placement

There are three ways you can pre-place instances in ACE:

- After the **Run Prepare** flow step, interactively pre-place instances using the ACE GUI's drag and drop placement features, or using 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 doing “`set_placement –fixed`” on that instance.

Using the "`set_placement -fixed`" commands in a PDC constraints file is recommended. If you use the location parameter in the user design RTL, then 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. First, it 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>.”. So you can look in your log file or TCL console and see messages for each instance placed with the location parameter. Next, the **Run Prepare** step applies all the `set_placement` commands in the PDC files. If you have a `set_placement` command in your PDC for an instance that is already placed with the location parameter, the PDC `set_placement` will override 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.

We recommend using 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 399) for details.
3. Then, save the pre-placement and automatically add it to the project (see Saving Pre-Placement Constraints (see page 403)).
4. Optionally, a pin assignment report can be generated for the current placement with the `report_pins` (see page 576) Tcl command.
5. Resume running the flow. The pre-placement data is used in the place and route solution.

Then, it is recommended to include the pre-placement constraints 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 215), the new pre-placement constraints files are enabled.
2. Then, 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 316), as indicated in the Flow View (see page 187). Timing analysis can be repeated with different Implementation (see page 312) options without having to re-run the rest of the flow, by double-clicking the appropriate **Run Timing Analysis** Flow Step. (see page 317)

The results of timing analysis are shown in a Timing Report (see page 322), which is automatically displayed as timing analysis completes. The most recently generated version of each timing report file is always available in every Implementation's `reports` sub-directory.

The active critical paths may also be viewed in the Critical Paths View (see page 174) and the Critical Path Diagram View (see page 170). Unlike the reports, the views only show the critical paths for the Active Project and Implementation (see page 316). 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 users may view a generated timing report from an Implementation's `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 322) is generated and displayed in the GUI whenever one of the **Run … Timing Analysis** Flow Steps (see page 317) 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` command.

Timing reports can be found in the implementation's (see page 312) reports directory, available for browsing via the Projects View (see page 238). In addition to the HTML report files displayed in the GUI, there are equivalent report files in text and csv (spreadsheet) formats.

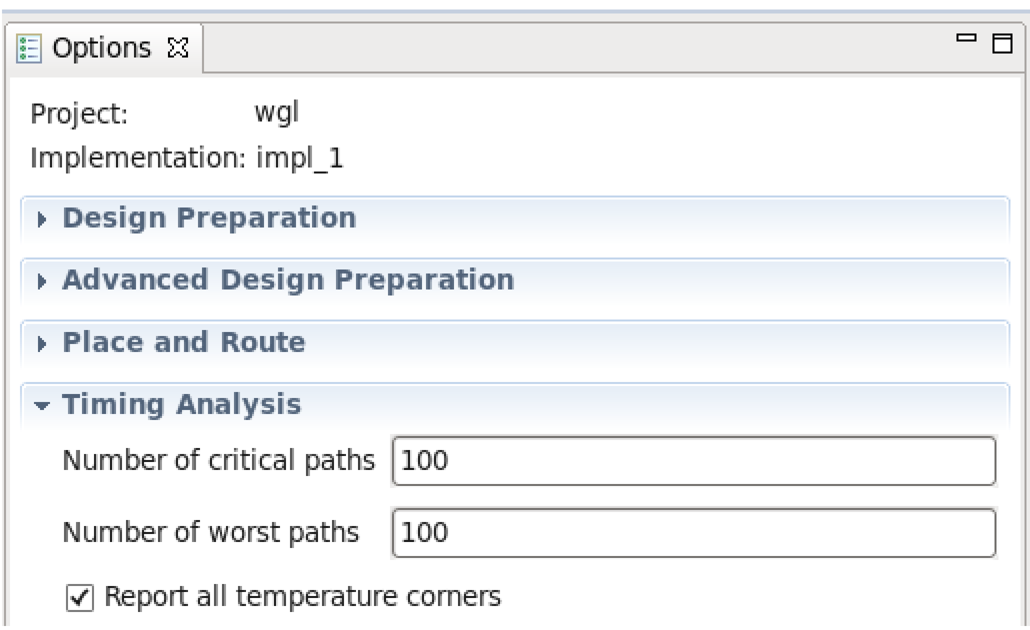
The Timing Analysis implementation options (see page) in the Options View (see page 215) determine how timing analysis is run and the amount of report information which will be generated.

Critical paths will also be displayed in the Critical Paths View (see page 174). The Path ID can be used to cross-reference between the Critical Paths view and the timing reports.

ACE also allows generating timing report across all temperature corners. ACE supports place and route across all temperature corners i.e. Place & Route will be optimized and timing reported across all temperature corners by enabling the "Report all temperature corners" under the Options (see page 215) --> Timing Analysis (please refer to Table:Timing Analysis Implementation Options under Options View (see page 215)).

The corresponding example ACE TCL command (see page 535) to set the required implementation option to enable this feature is: *set_impl_option -project "wgl" -impl "impl_1" "report_sweep_temperature_corners" "1"*

Figure 46: *Timing Analysis Options View*



Highlighting Critical Paths

> 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 178):

1. First, run one of the timing analysis flow steps to generate critical path data.
2. Then, in the Critical Paths View (see page 174), 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 174). 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 174) 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 249)) for easy access. Note that once they are in the ACE selection set, they will change their color to the selection color (the selection color overrides all other colors, including highlight colors).

To add a critical path to the current selection:

1. In the Critical Paths View (see page 174), click on the table row containing data for the desired path to select objects for.
2. To add the path to the current ACE selection set, click on the **Select Path** toolbar button ( ) on the Critical Paths View (see page 174) toolbar.

The path is now added to the selection in the Selection View (see page 249) and is shown with the selection color in the Floorplanner View (see page 178).

To add a critical path's pins to the current selection:

1. In the Critical Paths View (see page 174), click on the table row containing data for the desired path to select objects for.
2. To add the path's pins to the current ACE selection set, click on the **Select Pins** toolbar button ( ) on the Critical Paths View (see page 174) toolbar.

The pins are now added to the selection in the Selection View (see page 249) and are shown with the selection color in the Floorplanner View (see page 178).

To add a critical path's instances to the current selection:

1. In the Critical Paths View (see page 174), click on the table row containing data for the desired path to select objects for.
2. To add the path's instances to the current ACE selection set, click on the **Select Instances** toolbar button ( ) on the Critical Paths View (see page 174) toolbar.

The instances are now added to the selection in the Selection View (see page 249) and are shown with the selection color in the Floorplanner View (see page 178).

To add a critical path's nets to the current selection:

1. In the Critical Paths View (see page 174), click on the table row containing data for the desired path to select objects for.
2. To add the path's nets to the current ACE selection set, click on the **Select Nets** toolbar button ( ) on the Critical Paths View (see page 174) toolbar.

The nets are now added to the selection in the Selection View (see page 249) and are shown with the selection color in the Floorplanner View (see page 178).

Zooming to Critical Paths

To zoom the Floorplanner view (see page 178) to a critical path's region:

1. In the Critical Paths view (see page 174), click on the table row containing the desired critical path data.
2. To zoom to the path in the Floorplanner view (see page 178), click on the **Zoom to Path** toolbar button ( ) on the Critical Paths view (see page 174) 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 258) :

1. In the Critical Paths view (see page 174), click on the table row containing data for the desired path to print details for.
2. To print the details text, click on the **Print Path Details** toolbar button ( ) on the Critical Paths view (see page 174) toolbar.

Tip

Critical path details are also available in the timing report (see page 322).

Using Critical Path Diagrams

The Critical Path Diagram View (see page 170) provides a graphical representation of a single critical path; these paths are each selected from the table in the Critical Paths View (see page 174). The graphical representations will 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, make sure both the Critical Paths View (see page 174) and Critical Path Diagram View (see page 170) are visible. Then click on a row in the Critical Paths view's table. Now use the keyboard **Up** and **Down** arrow keys to change which row is selected in the table - the Critical Path Diagram view's diagram will be 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 and Arrows (and their supporting text) can vary depending upon the current settings in the diagram's fly-out palette (see page).

Nodes

Primary graph nodes (the larger circles in the diagram) 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 that some information will be hidden when the graph node (circle) is too small to contain it; to see all enabled information, the diagram will have to 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. They too can display various pieces of information that may be enabled and disabled via the fly-out palette. In addition to the arrow's direction, 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 their cumulative ps of **Delays**. **Net Names** and **Fanouts** are never displayed for bold arrows, since they make no sense in this context.

Skinnier arrows will be shown when the **Intermediate Nodes** fly-out palette setting is enabled - these will each represent an individual net. Because these skinnier arrow each represent individual nets connecting the instances, the individual nets' **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 174)'s 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

Once the graph has helped the user track down which nets and/or instances they wish to tweak for timing purposes, it can be helpful to find those objects in the Floorplanner View (see page 178). To do so, the user may use the techniques described in Selecting Critical Path Objects (see page 406), or they can 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 249) to cause the Floorplanner View (see page 178) 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 must be done in 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 174), click on the **Save Script File** toolbar button ( ) to open the Save Script File dialog (see page 284).
2. Enter a valid file in the Save Script File dialog (see page 284) and click **Save**.
3. Source the saved script file from within the synthesis tool's Tcl prompt.
4. After the file is sourced, open the synthesis tool's schematic viewer. At the Tcl prompt, type "`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 322).

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 596) 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 242), or use the display_properties (see page 545), get_properties (see page 564), or get_property (see page 564) 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 (under the covers) the `acx_stapl_player` to interact with an Achronix FPGA. The JTAG interface is used by the Download View (see page 176), Snapshot Debugger View (see page 253), JTAG Browser View (see page 199), and the HW Demo View (see page 189). The automated SerDes Link Tuning will also use JTAG.

> For more details on managing the physical connection between the workstation, the Bitporter pod or FTDI FT2232H device, and the FPGA board, see the *Bitstream Programming and Debug Interface 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 189).

Configuring the DCC Connection

The ACE DCC (Demo Command and Control) connection is how the ACE software communicates with demo and/or reference designs running on Achronix hardware. The DCC connection is managed using the Configure DCC Connection Preference Page (see page 288).

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 for "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)

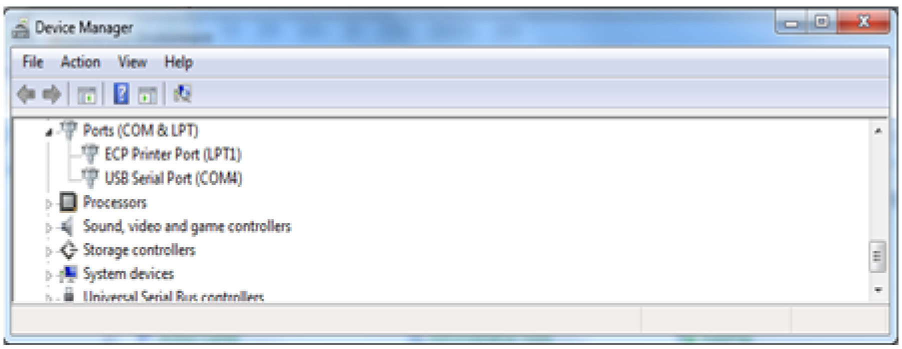
The necessary FTDI USB drivers are already included in supported Linux distros. If the user's distro-supplied USB drivers do not work correctly, the necessary drivers may be downloaded from the FTDI website. Contact Achronix support if you need help finding the Linux drivers.

Configuring the USB drivers

The DCC cable USB drivers will automatically choose a serial port to be redirected to the USB cable. Users may need 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 COM port we want will be shown as "USB Serial Port (COM*)", with the actual COM port number included.



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's serial port assignment.

Configuring ACE

Once the user knows 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 288) (**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 will vary according to the operating system in use.

In Windows, this will be a COM port, typically one of `COM1` through `COM9`. `COM3` is the most frequent (and thus default) choice.
In Linux, the serial ports will be named `/dev/ttyS*` with the `'*'` being replaced by a number. This will typically be 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 user site may only have a single JTAG Programmer Device connected to a JTAG scan chain containing a single Achronix FPGA, many sites will have multiple JTAG Programmer Devices, and/or multiple devices within the connected JTAG scan chain. For these reasons, the ACE tools need to be able to specify both which JTAG Programmer Device Connection to use, and which JTAG scan chain member is relevant.

The various tools within ACE will pull 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's Configure JTAG Connection Preference Page (see page 288)
- The Implementation Options (see page 312), as managed within the Options View (see page 215). (Note that the implementation options for the JTAG scan chain values get saved in the STAPL `*.jam` bitstream file during the **Run Bitstream Generation** Flow Step (see page 317), when that file is being created.)
- Command-line overrides

The GUI Views responsible for live FPGA interactions will retrieve their JTAG Programmer Device and JTAG connection details from the Configure JTAG Connection Preference Page (see page 288). Tools that generate a new bitstream for an implementation (typically during the Flow (see page 316), but some may also be triggered ad-hoc) will retrieve the JTAG Scan Chain values from the appropriate Implementation Options (see page 312) as managed by the Options View (see page 215) in the "Bitstream Generation" group; these JTAG Scan Chain values are then saved in the STAPL `*.jam` bitstream file for later use during downloads. The various STAPL / download tools may take their settings from the preferences, may take them from the current implementation options, may take them from the STAPL `*.jam` file, or may take them from the command-line, depending upon how the tool is typically used.

> 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's shell / command-line. The use of this tool is covered in detail within the *Bitstream Programming and Debug Interface User Guide (UG004)*.

> **Special Note For Sites With More Than One Connected FPGA**
>
> The interactive members of the ACE tools currently assume that only a single FPGA will be of interest to a user at a time, and those interactive tools all share a single configuration for the user on the Configure JTAG Connection Preference Page (see page 288). 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, users must remember to change the JTAG Programmer Device and JTAG scan chain values stored on that preference page every time they wish to alternate between FPGAs.

JTAG Programmer Device Connection

It is often possible that multiple JTAG Programmer Devices are visible from a single workstation. Because of this, users must configure ACE software to know which JTAG Programmer Device is intended to be used. The GUI's JTAG Programmer Device and JTAG connection details are managed primarily through the Configure JTAG Connection Preference Page (see page 288) (for interactive GUI tools) and the Implementation Options within the Options View (see page 215) (for tools called within the Flow (see page 316)).

> **Bitporter pods may be damaged if improperly connected or power-cycled!**
>
> Users must read the *Bitstream Programming and Debug Interface User Guide (UG004)*, as well the user guide (s) specific to their development kit, before physically connecting the Bitporter's 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 *Bitstream Programming and Debug Interface User Guide (UG004)*, as well the user guide(s) specific to the development kit, and any related release notes.

The *Bitstream Programming and Debug Interface User Guide* (UG004) will also cover 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` is able to automatically detect 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 told which of the JTAG Programmer Devices (by name) should be used, STAPL program execution will fail, because ACE doesn't dare pick a JTAG device randomly.

It is strongly recommended that ACE users choose to specify their chosen JTAG Programmer Device by name in all cases, instead of relying upon auto-detection. Not only will this avoid problems if additional JTAG Programmer Devices are connected at a later date, but connections to named devices will be faster to initiate.

Performance Tip

Users can save several seconds of initialization time on every JTAG connection if their JTAG Programmer Device is specified by name instead of using auto-detection.

The details of JTAG device naming are covered thoroughly in the *Bitstream Programming and Debug Interface 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 will use the prefix of 'usb' and be addressed by their serial number (which should be visible on a sticker on the pod itself), like '`usb12345`' for a serial number of 12345. Bitporter pods connected over Ethernet will use the 'net' prefix and their IP address as the suffix, like '`net192.168.1.123`' for an IP address of 192.168.1.123.

JTAG Scan Chain

The 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 288).

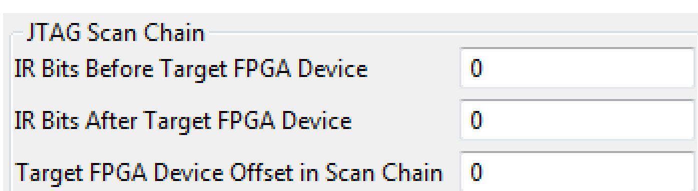


Figure 47: ***JTAG Scan Chain Fields, from the Configure JTAG Connection Preference Page***

Note

The default JTAG scan chain preference values, all zeros, is always correct for single-device scan chains. For multi-device scan chains, the defaults of all zeros will never work.

When multiple FPGA devices are attached to the same JTAG scan chain, the user must specify which FPGA device is the target. Because different FPGA devices have different instruction sizes, the total instruction length before and after the target must be specified as well.

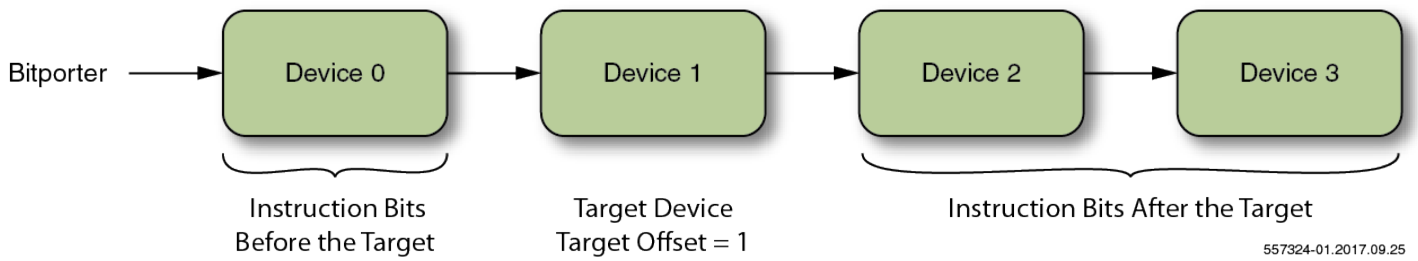


Figure 48: *Multi-device Scan Chain with Bitporter Example*

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's 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 will not be scanned through the target FPGA. Instruction bits after the target's instruction bits will be scanned through the target FPGA before arriving at their scan chain destination. The key detail is that ACE thinks of 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 167: *Example scan chain values for a series of four Achronix FPGA devices 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's 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.

Instruction Register Lengths Vary by Vendor and Device

For users new to JTAG, it may be worth mentioning that if non-Achronix devices are in the scan chain, it is extremely likely that their Instruction Registers will not be 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 will always each have a DR length of one.

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 may help clarify the relationships:

Table 168: *STAPL vs. ACE Terminology Differences Regarding 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.

Details on the Snapshot hardware architecture 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 FTP site (login required), or from an Achronix FAE.

The Snapshot content here in the ACE User Guide will provide a general overview of functionality which is common to all Achronix devices, and is focused on the Snapshot Debugger's user interface for real time in-system debugging.

Snapshot Design Flow

The JTAG connection must be configured before using the Snapshot Debugger!

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 Snapshot Debugger view. The configuration is managed using the Configure JTAG Connection Preference Page (see page 288), which is easily accessible by pressing the **Configure JTAG Interface** ( ) button in the Snapshot Debugger view. See Configuring the JTAG Connection (see page 411) 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's design in real-time, and optionally send stimuli to the user's 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 user will be able to debug the design in the ACE GUI using the Snapshot Debugger view (see page 253) and the VCD Waveform Editor (see page 27), found within the Bitporter perspective (see page 23).

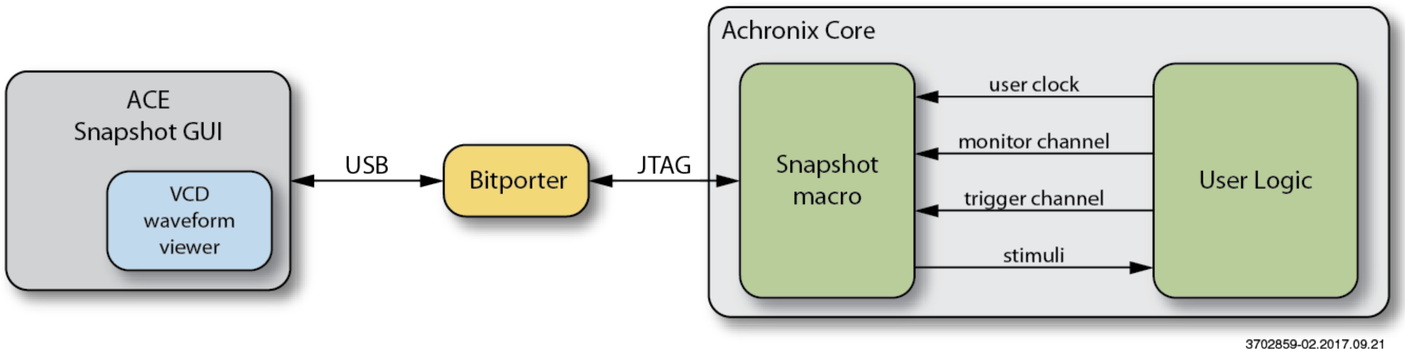


Figure 49: ***Snapshot Communication with the Snapshot Debugger View within ACE (Running on the Host PC)***

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 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 a designer to configure 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 will be displayed in the ACE Editor Area using the VCD Waveform Editor (see page 27). The VCD output can also be read into a third-party waveform viewer.

At a high level, to utilize Snapshot the user must first:

1. instantiate the Snapshot macro `ACX_SNAPSHOT` in the user's design
2. synthesize the design
3. place and route the design in ACE
4. generate the Bitstream for the design in ACE
5. configure ACE's JTAG connection to the FPGA (see Configuring the JTAG Connection (see page 411))
6. program the Achronix device with the Bitstream
    - use of the ACE GUI's Download View (see page 176) is documented in the section Playing a STAPL File (Programming a Device) (see page 432)
    - use of the `acx_stapl_player` executable on the command-line is documented in the *Bitstream Programming and Debug Interface User Guide* (UG004)

Once those prerequisite steps are complete, the ACE GUI's Snapshot Debugger View (see page 253) allows the user to evaluate/interact with the running design in real-time

The following sections will further explain Snapshot and guide the user through the process.

Accessing the Snapshot Debugger

Open the ACE GUI and Select Your Project

Open the ACE GUI tool, and load or activate your project in the Projects View as shown below. See the Loading Projects, (see page 353) Setting the Active Implementation (see page 360), and Working with Projects and Implementations (see page 351) sections for details.

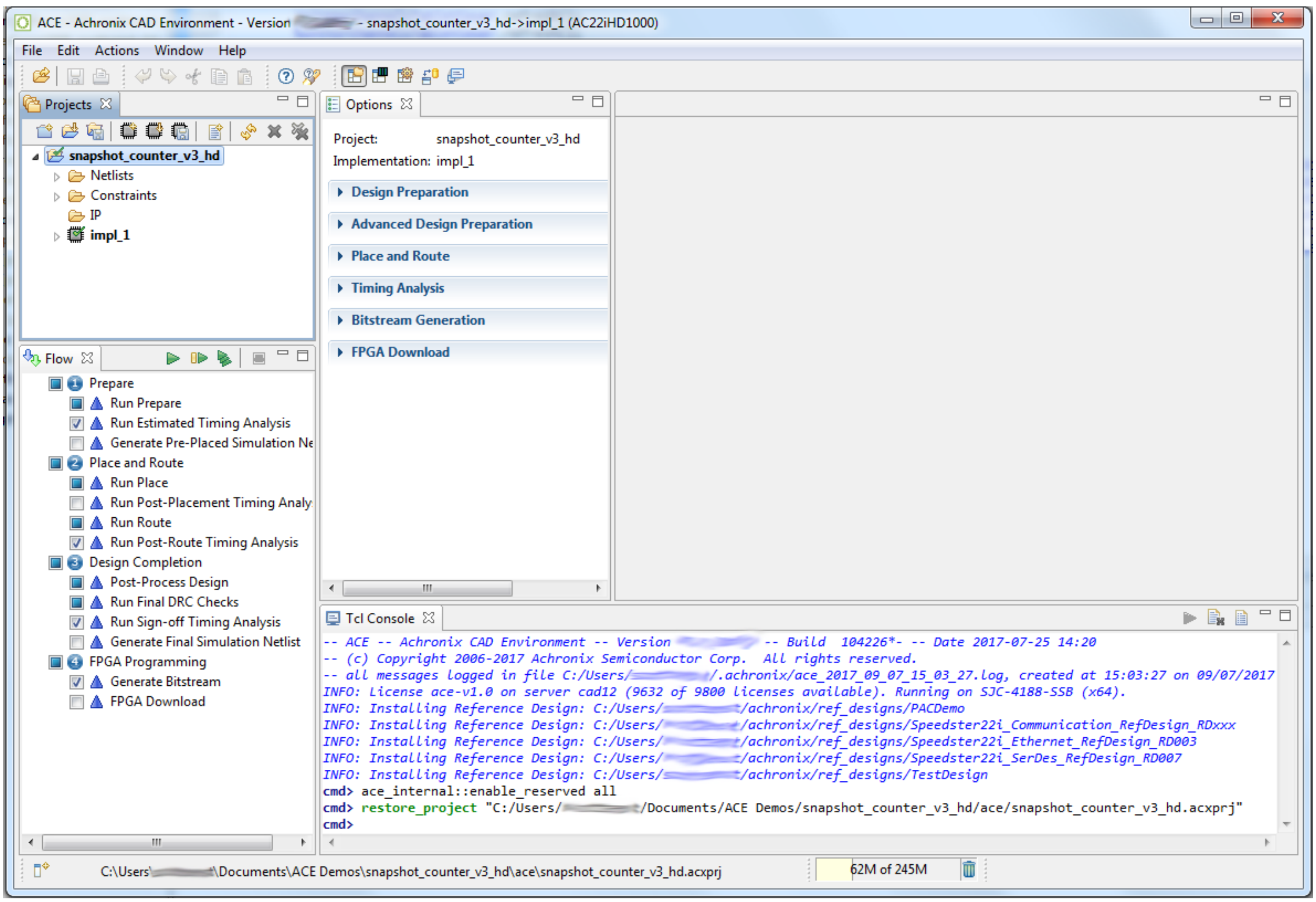


Figure 50: *ACE Tool*

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 345) section. The Snapshot Debugger View (see page 253) should be visible by default, as shown below. If not, you can click **Window→Show View→Snapshot Debugger** from the main menu bar.

The Snapshot Debugger View (see page 253) should have automatically loaded the default Snapshot configuration file for your project that was generated when you ran your design through place and route, located in `<ace_project_dir>/<active_impl_dir>/output/names.snapshot`. If it loaded, you will see the correct signal names from your user design in the Trigger Channels, Monitor Channels, and Stimuli tables. If it did not automatically load, you can click the **Load Snapshot Configuration** ( ) toolbar button in the Snapshot Debugger View (see page 253) to browse to the location of you preferred `*.snapshot` configuration file, or manually enter the signal names, channel widths, etc to match your design.

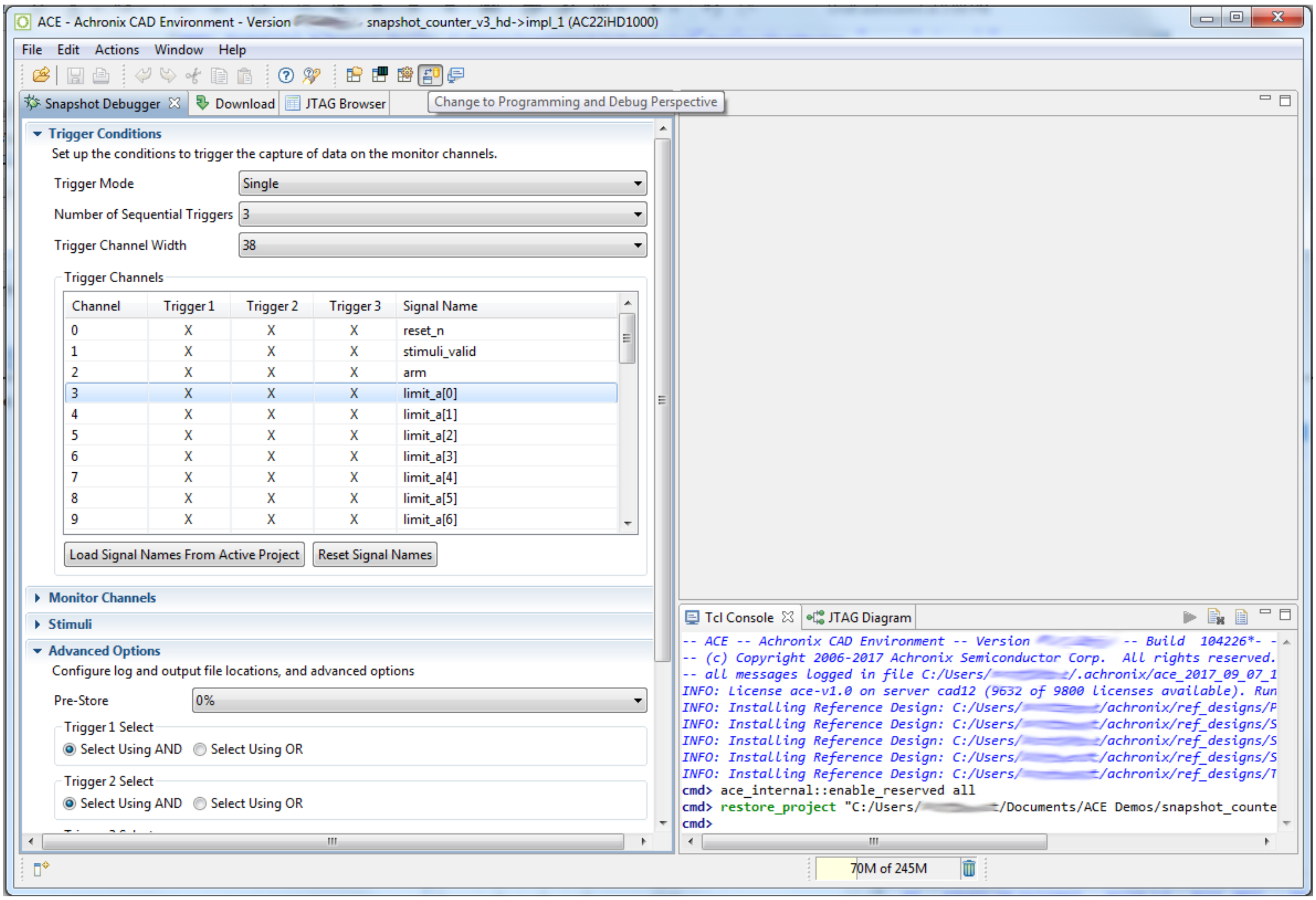


Figure 51: *Snapshot Debugger View*

Configuring the Trigger Pattern

> 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 317)), 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 253). 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).

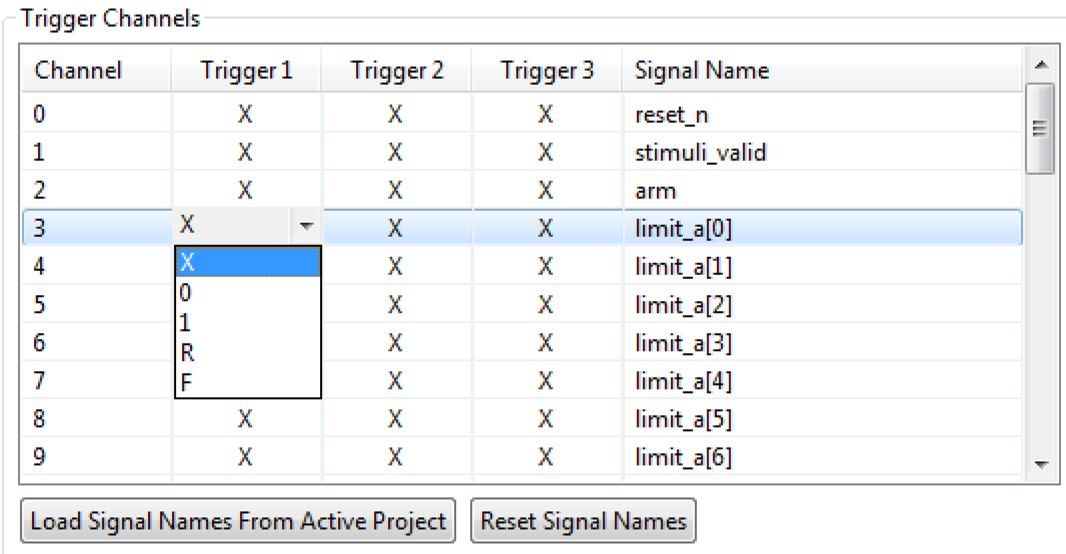
> ⚠ If any active Trigger is configured with as all X's (don't care), the trigger pattern will be a match on the first clock cycle that trigger is evaluated.

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

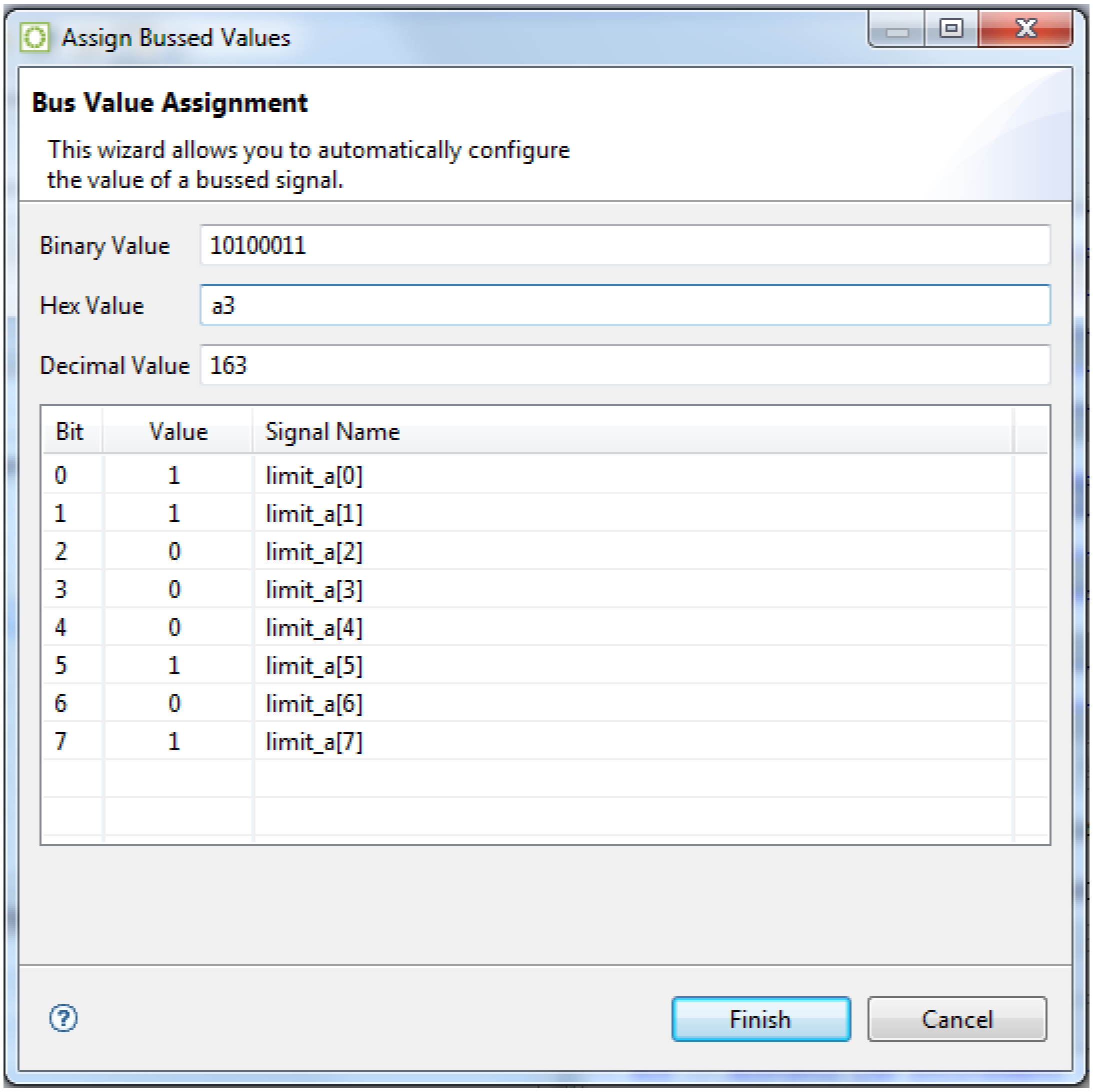
For each channel, a value of **X** (don't care), **R** (rising edge), **F** (falling edge), **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under each "Trigger" column as shown below.

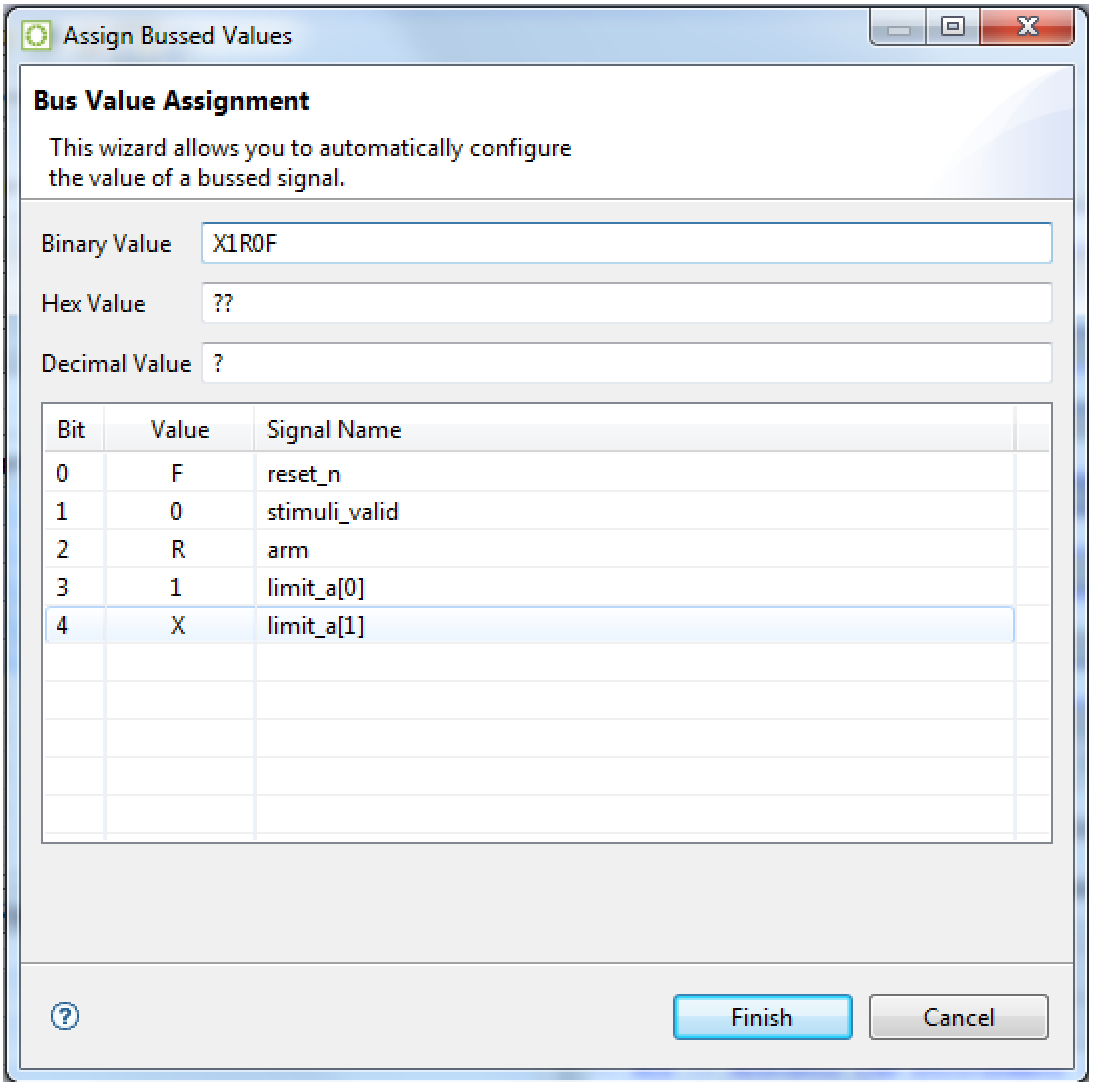


Setting Multiple Pattern Values as a Bus

The Assign Bussed Values Dialog wizard allows the user to assign a value to multiple signals from the SnapShot Debugger view's (see page 253) "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 253). 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 253) 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 253) 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.


Assign Bussed Values
Bus Value Assignment
This wizard allows you to automatically configure
the value of a bussed signal.
Binary Value
10100011
Hex Value
a3
Decimal Value
163
Bit
Value
Signal Name
0
1
limit_a[0]
1
1
limit_a[1]
2
0
limit_a[2]
3
0
limit_a[3]
4
0
limit_a[4]
5
1
limit_a[5]
6
0
limit_a[6]
7
1
limit_a[7]
Finish
Cancel



See Assign Bussed Values Dialog (see page 264) for more information on this dialog.

Configuring the Monitor Signals

> The Monitor Signals 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 317)), 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 253) 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 will be the width of the *i_monitor* bus.)

The value of **Number of Samples** in the SnapShot Debugger view (see page 253) 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 will be used and a warning will be printed in 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 "Monitor Channels" table are then used as labels on the captured signal data in the VCD waveform output, and will be visible in the VCD Waveform Editor (see page 27).

Multiple signals can be combined into a bus by selecting multiple rows in the "Monitor Channels" table, right-clicking on 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 262). After configuring the bus in the Assign Bussed Signal Names Dialog (see page 262), the bus name and indices are propagated to all the previously-selected signals. To select a contiguous range of rows, select the first signal, hold the Shift key, and select the last signal. To select a non-contiguous set of rows, select the first signal, then while holding down the Ctrl key on the keyboard, select the other signals.

Signal names may be returned to their defaults by selecting the **Reset Signal Names** button under the "Monitor Channels" table. Note that this 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 317)), 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 317)), 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.

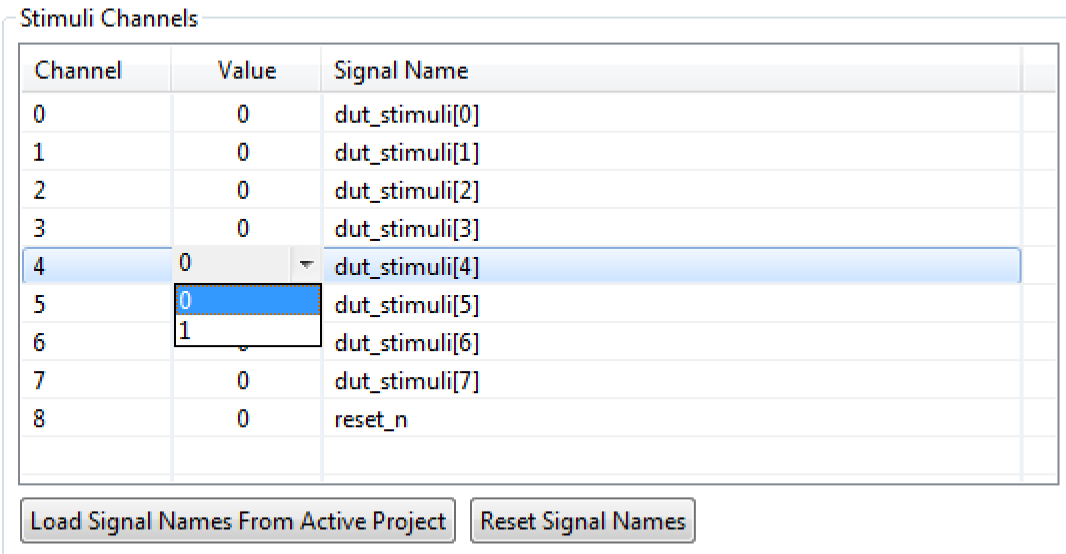
This *o_stimuli* output is optional, and need not be connected to the DUT - it may safely be left floating when the user wants to use Snapshot to only read signals.

The value of **Stimuli Channel Width** in the SnapShot Debugger view (see page 253) 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 will be 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

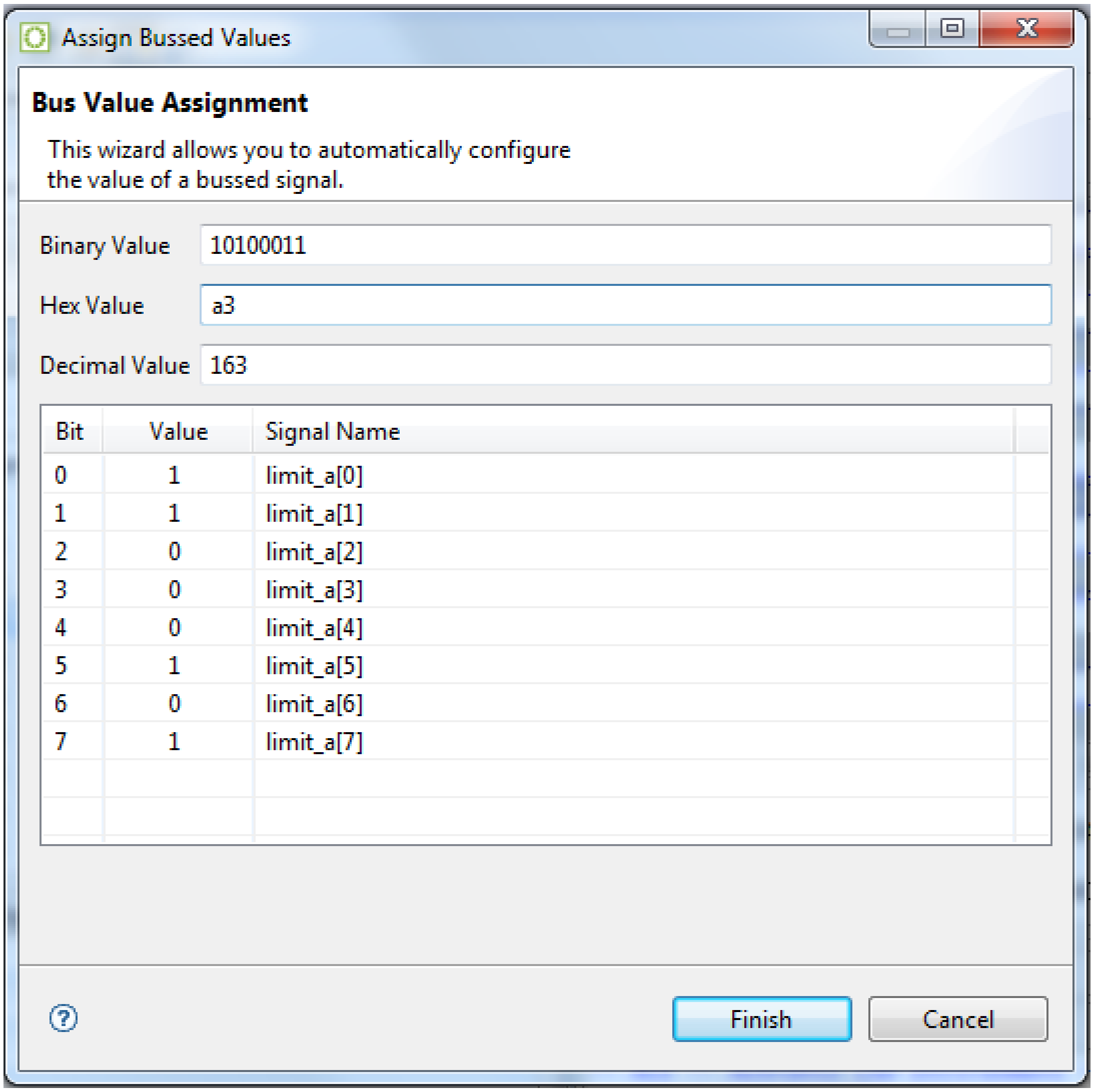
For each channel, an output value of **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under the "Value" column as shown below.



Setting Multiple Stimuli Values as a Bus

The Assign Bussed Values Dialog wizard allows the user to assign a value to multiple signals from the SnapShot Debugger view's (see page 253) "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 253). 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 253) 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 253) 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.



See Assign Bussed Values Dialog (see page 264) for more information on this dialog.

Configuring Advanced Options

Pre-Store

In the Snapshot Debugger View (see page 253), 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 the user has configured Snapshot to use a monitor depth of 1024 samples. See the table below:

Table 169: ***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

"Pre-Store" value	Samples collected before trigger	Samples collected after trigger
50%	512	512
75%	768	256

When a **Pre-Store** value other than **0%** is selected, the `.vcd` file will contain a signal *snapshot_pre_store* that transitions (goes low) at the point where the (last sequential) trigger event occurred. Thus, users may easily find the trigger event 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, 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.

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 will be 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) will determine 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 will 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**'s directory or to the **Working Directory**. (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 from the default. (The default will be `'<active_impl_dir>/log/snapshot.log'`, or if there is no Active Project and Implementation (see page 316), then `'<user_home>/snapshot.log'`.) If an error occurs during setup or while reading back the sample information, the Snapshot log file will contain 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 from the default. (The default will be `'<active_impl_dir>/output/snapshot.vcd'`, or if there is no active implementation, then `'<user_home>/snapshot.vcd'`.)

Collecting Samples of the User Design

Using the Startup Trigger

The Startup Trigger feature requires that the end user has configured the initial startup trigger parameters 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 will connect to the running ACX_SNAPSHOT circuit over JTAG and wait for the startup trigger condition to be met, retrieve the trace buffer contents, and output a VCD file. This feature is useful to capture trigger events that happen very soon after the Achronix FPGA enters user mode. Once the **Arm Snapshot** ( ) button is pressed, 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

Once all the fields in the Snapshot Debugger View (see page 253) are configured, and the design is running on the target device, Snapshot is ready to be Armed.

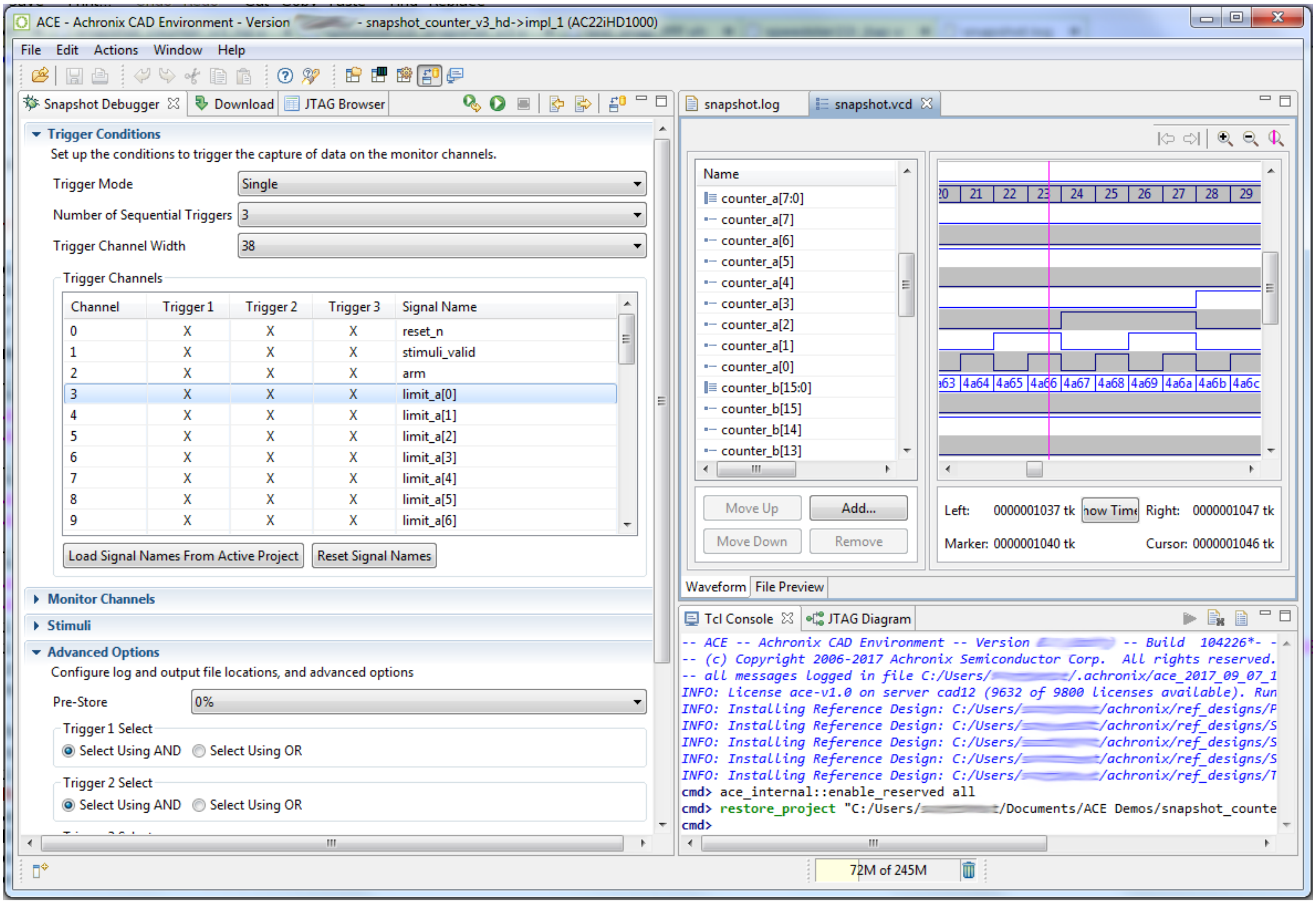
Select the **Arm** button [or the **Arm Snapshot** ( ) button in the SnapShot Debugger view's toolbar], and the ACE Snapshot Debugger will send the configuration data (including the optional *Stimulus*) to the ACX_SNAPSHOT circuit running on the Achronix device, wait for the trigger condition(s) to be met, retrieve the trace buffer contents, and output a VCD file as well as a LOG file.

Once Armed, Snapshot will begin 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 will open in an ACE Text Editor (see page 26), while the waveform (`.vcd`) file will open in the ACE VCD Waveform Editor (see page 27).) If an error occurs during Snapshot Debugger configuration or while reading back the sampled information (trace buffer), the Snapshot Log file will contain the relevant error messages, and the Snapshot Waveform file will not be created/updated.

The **Cancel** ( ) button aborts the Snapshot Arming process. The Snapshot Log file will be updated, but the Snapshot Waveform file will not be created/updated if Cancel is pressed. Cancel is useful if you accidentally send in trigger conditions that are never matched.

If using **Repetitive** Trigger Mode, Snapshot will repetitively execute the Arm action for the number of records specified, or until Cancel is pressed. See Configuring the Trigger Pattern (see page 419) for details on the Repetitive Trigger feature.



Saving/Loading Snapshot Configurations

Users may wish to re-use an existing known-good Snapshot configuration (the collection of settings in the Snapshot Debugger View (see page 253)) 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's (see page 253) toolbar.

These Snapshot configurations may then be loaded later by using the **Load SnapShot Configuration** ( ) button, found in the Snapshot Debugger View's (see page 253) toolbar.

> Previously saved Snapshot configuration files are necessary to run Snapshot in Batch Mode (see page 430).

> When a user design containing the `ACX_SNAPSHOT` macro completes the Flow Step (see page 317) **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 316) is available, the Snapshot Debugger View automatically loads the implementation's `names.snapshot` file to pre-populate the relevant fields of the view. Note that 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 430) 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 429) 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 be configuring the Snapshot Debugger View (see page 253) in the ACE GUI and saving the Snapshot configuration (see page 429).

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)

> **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 288), which is easily accessible by pressing the **Configure JTAG Interface** ( ) button in the Download view. See Configuring the JTAG Connection (see page 411) for more details.

A STAPL† bitstream file (`*.jam`) can be run or played from the Download View (see page 176). 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 176).

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 file extension `.jam`.) If that default file does not exist, the "STAPL Actions and Procedures" tables will display an appropriate error message.

Selecting the **Manual Selection** option allows the user to manually choose the path to a desired STAPL file from an arbitrary location. The **Browse** button can be used to facilitate file selection, the user can type a new full path, the user can edit an existing path, or the user can choose from previously used `*.jam` files via the editable drop-down combo-box. (Press the down arrow on the right of the combo-box to see a list of previously used files.)

Lab Mode

When ACE is in Lab Mode, it is impossible to load designs, so there will never be a "current" design and implementation. Thus the option **Default File From Current Design/Impl** will be disabled, and the user will be forced to use **Manual**

Selection, and browse or type the STAPL design file to be used. The filepath combo will still retain the last 15 files used previously, 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 will be played/run. Deselected Procedures will be skipped.

Table 170: *STAPL Procedure Execution States*

State	Icon	Description
(default)		These required procedures are always selected, and will always be executed. The user is not allowed to disable these procedures.
Recommended		These procedures are initially enabled, but users are allowed to disable them.
Optional		These procedures are initially disabled, but users are allowed to enable them.

The checkbox icon appearance will vary 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 258) and saved in the ACE log file. Additionally, a log file of just the run itself is opened for viewing.

When programming the FPGA using the Download view, the JTAG Scan Chain configuration specified on the Configure JTAG Connection Preference Page (see page 288) will override 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 215).)

When using the `acx_stapl_player` from the command-line (instead of through the Download view's GUI), the JTAG Scan Chain configuration embedded in the STAPL file will be used instead, unless overridden with command-line arguments.

Optimizing a Design

There are numerous methods of design optimization available to ACE users.

Many optimizations are able to 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), which may be configured from the Options View (see page 215) and/or the Tcl command `set_impl_option`.

Achronix optimization experts have also collected together into Option Sets (see page) the implementation options which are known to work well together. These option sets may be used to create new implementations for user designs, allowing users to compare/contrast how various optimizations affect their achieved frequencies and required runtimes.

Other optimizations must be performed manually by the user, typically by editing the design's source RTL or `.sdc` timing constraints. Analyzing Critical Paths (see page 404) 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 363), the Multiprocess View (see page 204) allows users to generate new implementations with Achronix-selected combinations of Implementation Options (see page). These known-good subsets of implementation options are called Option Sets (see page). Achronix optimization experts have collected several option sets which are known to result in increases for the achieved frequencies in a variety of representative designs.

The Multiprocess view will allow users to select a starting template implementation (see page 312), and will then generate new implementations using the template implementation as a base. Each generated implementation will override 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 will be left with the same settings as existed in the template implementation. Only the options specified in the option set will be overridden to take on new values. The newly generated implementation will be given a name which includes the option set name for clarity. (The generated name will be the template implementation's name as a prefix, with the option set's name as the suffix.)

Please see the information in Running Multiple Flows in Parallel (see page 363), which discusses the basic use of the Multiprocess View (see page 204) and Multiprocess Summary Report (see page 323) - the rest of this section will build upon those descriptions.

Selecting the Implementations to be Generated and Run in Parallel

After Finding the Multiprocess View (see page), Configuring the Execution Queues (see page), and Configuring the Desired Flow to be Followed by the Selected Implementations (see page), the user is ready to select the implementations to be generated.

First, select the radio button labeled **Generate Implementations from Option Sets**, found within the Multiprocess view's "Select Implementations (see page)" section.

Then, in the Projects View (see page 238), select (activate) the desired project (see page 312) and implementation (see page 312). The Implementation Table within the Multiprocess view's "Select Implementations (see page)" section will then be updated to display a collection of implementations based upon the active implementation (see page 316).

The first entry in the Implementation Table will be the active implementation itself. This implementation will be the template from which all the generated implementations will be derived. All other implementations in the table will be 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 will indicate briefly what implementation option changes are caused by each option set (thus describing how each generated implementation will differ 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) commences, ACE will:

1. remove implementations in the active project with the same name as to-be-generated implementations
2. create new implementations (exact copies of the template implementation) with the required names

3. apply 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. add 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 363), starting from Starting Background Execution (see page).

Each generated implementation which is selected will overwrite *without prompting* any already-existing implementation with the same name in the active project. The template (active) implementation will not be changed/overwritten. If the user wishes to keep a previously-existing implementation with a to-be-generated name collision, 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), the final step of an optimization pass is usually to compare the results and choose which generated option set implementation provides the best QOR in comparison to the template implementation.

That best generated implementation could then be renamed, so that it doesn't get overwritten by future multiprocess runs. (For example, 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 will not be inherited by the option set implementations.

Also, there is scenario where all multiprocess results can be identical. The cause and a workaround are described in Running Multiple Flows in Parallel (see page 371).

Placement Regions and Placement Region Constraints

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.

ACE automated placement often produces better QOR than user-defined placement regions/constraints

If users choose to attempt to use Placement Regions and Placement Region Constraints, it is highly recommended that users also keep a parallel implementation of their project lacking the user-defined Placement Regions. 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 the user 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 328) 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 doesn't 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 178) and Placement Regions View (see page 234). The Search View (see page 245), Selection View (see page 249), Critical Paths View (see page 174), and Netlist Browser View (see page 211) may also be used to assign instance placement constraints by using drag-and-drop operations.

Users should 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 385), nor are they directly searchable in the Search View (see page 245) or with the Tcl `find` (see page 552) command.

Placement Region Preferences

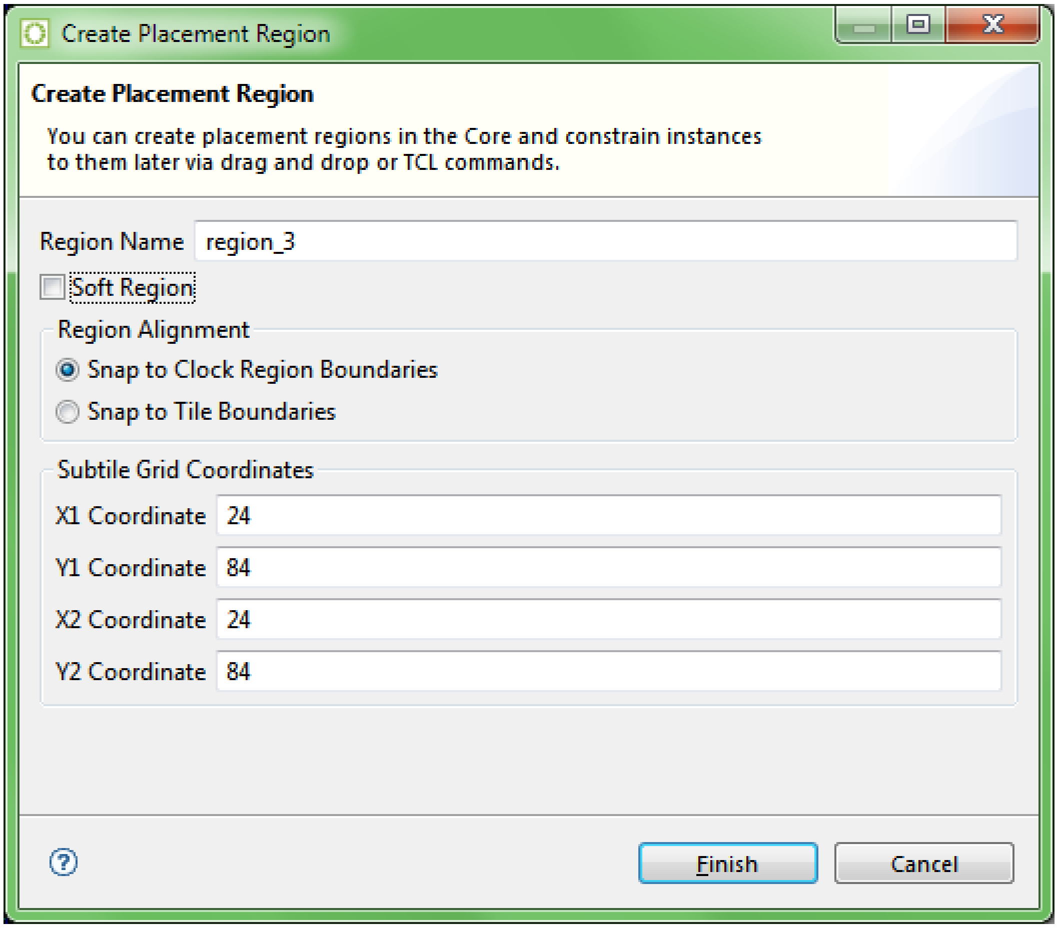
There are a number of user preferences which may be configured to alter how the mouse creates Placement Regions and assigns Placement Region constraints. These preferences are found on the Placement Regions Preference Page (see page 305) .

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's Placement Region Tool () is active.
2. (Optional) If the Placement Region is meant to align with one or more Clock Regions (see page 328), enable the overlays for those regions from the Clock Regions View (see page 167). This will not affect the functionality in any way, but will make it easier to know where to define the region bounds.
3. Press and hold the left mouse button 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 mouse to the opposite corner of the desired Placement Region area. Release the left mouse button when the mouse reaches the desired location.
5. ACE calculates the enclosed subtile grid coordinates, growing as necessary to ensure all partially-enclosed subtiles are fully enclosed.
6. The Create Placement Region Dialog (see page 271) pops up pre-populated with the calculated subtile coordinates.

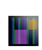


7. Fill in the desired Placement Region name.
8. Select whether the Placement Region should be based upon the simpler Clock Regions, or the more granular subtiles.
9. Press the **Finish** button to create the new Placement Region.
10. ACE adds the new Placement Region to the table in the Placement Regions View (see page 234) and displays it as a translucent overlay within the Floorplanner. (At this point, the region will contain no constraints.)

Resizing an existing placement region

Existing Placement Regions may be resized with the Tcl command `set_region_bounds`, or with the mouse in the Floorplanner view. Any existing Placement Region Constraints for that region will be kept - only the enclosed area will be updated.

To resize a Placement Region with the mouse in the Floorplanner View (see page 178):

1. Ensure the Floorplanner's Placement Region Tool () is active.
2. In the Placement Regions View (see page 234), ensure the checkbox in the first column is selected for the desired Placement Region. This will make the Placement Region overlay visible within the Floorplanner view.
3. Ensure the **Snap To:** option in the Placement Regions Preference Page (see page 305) is configured as desired.
4. (Optional) If the Placement Region is meant to align with (snap to) one or more Clock Regions (see page 328), enable the overlay for those regions from the Clock Regions View (see page 167). This will not affect the functionality during the resize in any way, but will make it easier to know where to define the region bounds.

5. Move the mouse over either the upper-left corner or the lower-right corner of the placement region to be resized. The mouse cursor will change to a diagonal resize cursor when the mouse is 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's 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 Tcl command `set_region_bounds`, or with the mouse in the Floorplanner View (see page 178). 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 may 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. Users will frequently desire 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's 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 will make the Placement Region visible within the Floorplanner view.
3. Ensure the "Snap To" option in the Placement Regions Preference Page (see page 305) is configured as desired.
4. (Optional) If the Placement Region is meant to align with (snap to) one or more Clock Regions (see page 328), enable the overlay for those regions from the Clock Regions view (see page 167). This will not affect the functionality during the resize in any way, but will make it easier to know where to define the region bounds.
5. Move the mouse over the placement region to be moved. The mouse pointer will change to a move cursor when the mouse is over any placement region.
6. Press and hold the left mouse button and drag 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's 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 commands`, or interactively with drag-and-drop mouse actions in the ACE GUI.

When using the add_region_insts (see page 536) or add_region_find_insts (see page 536) commands you may specify the instances to constrain 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 (see page 552) 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 the instance list is specified with a find (see page 552) 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 (see page 536) command is evaluated (which happens at the beginning of the run_prepare (see page 585) flow step), not at the time at which it is applied with the apply_placement (see page 537) command (which happens at the end of the run_prepare (see page 585) flow step). New instances which may be created during run_prepare (see page 585), even if they would have matched the command/expression, will not be included. Therefore, the add_region_insts (see page 536) command is best reserved for interactive use. The add_region_find_insts (see page 536) command, on the other hand, specifies the find (see page 552) command as a string argument to be batched and evaluated later during apply_placement (see page 537). Therefore, the recommended practice is to use the add_region_find_insts (see page 536) command instead of add_region_insts (see page 536) when writing PDC constraint files.

```
add_region_find_insts "region_1" "find -insts inst*"
add_region_find_insts "region_1" "find -insts inst* -filter {@type=DFF && @clock_domain=clk1}"
```

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 (see page 536) 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. Users may want to create more concise constraints for their design 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 will be excluded. (Setting these inclusion/exclusion preferences for mouse actions is done on the Placement Regions Preference Page (see page 305).)

```
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 will be 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 will be 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 will be updated.

Overlapped Placement Regions

If multiple placement regions overlap visibly in the Floorplanner view, any Instances dropped within the visibly overlapping area will be ignored. In such cases, the user must either drop in the Placement Regions view, or drop in the Floorplanner view where there is no visible overlap. (Users may disable Placement Region overlays from the Placement Regions view to eliminate visible overlaps - in these cases, constraint assignment will occur 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 234), 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 will turn red to indicate the problem.

To view a list in the Tcl Console View (see page 258) of all objects constrained to a placement region, the user may use the Tcl command `get_region_insts`, or the user may right-click the mouse upon the desired Placement Region in the Placement Regions view, and then select (left-click) "Print Instances".

Removing a placement region constraint from an object

Users may remove placement region constraints from individual Core/Boundary Instances, or from all instances assigned to a region at once.

To unassign a placement region constraint for individual core instances, use the Tcl command `remove_region_insts`.

To remove all instance constraints from a placement region, use the same Tcl command, or in the Placement Regions View (see page 234), right-click the mouse on the desired placement region, and select (left-click) "Clear Placement Region".

Saving placement region definitions and placement region constraints

The user may save placement region constraints from the Floorplanner View (see page 178) with the "Save Pre-placement Constraints" action (which displays the Save Placement Dialog (see page 281)), from the Placement Regions View (see page 234) with the "Save Placement Regions" action (which displays the Save Placement Regions Dialog (see page 284)), or by using the Tcl command `save_regions` directly.

> **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 user may edit the saved PDC file 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 234) by right-clicking the region in the table and selecting "Remove Placement Region".

Alternately, the Tcl command `remove_region` 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 will 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) will be packaged into and delivered in a single tarball, ZIP, or Windows installer file, downloadable from the Achronix FTP site. A set of installation instructions will be provided as a separate document, (not here,) as the details may vary for each design. Installation may 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 hacked / played with, while Demo designs are black boxes. Reference designs will typically be installed into the user's 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 will use the same framework within ACE, and both will be 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

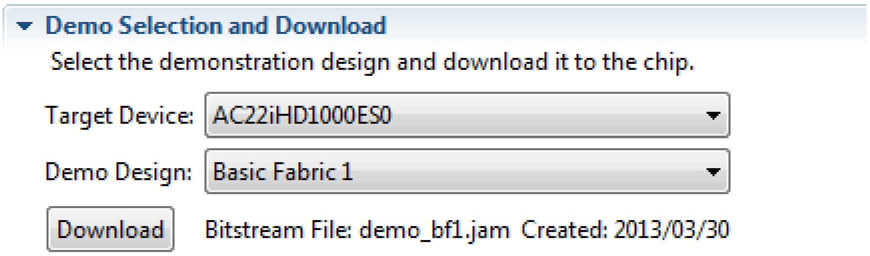
By default, when ACE starts up, it will search for installed HW Demos in the following paths:

- `<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.

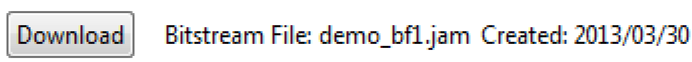
Selecting The Target Device And Demo



At the top of the HW Demo View (see page 189) are controls for selecting the target device and an associated demonstration design. Once you have selected 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.

Note that if no demos are installed, these controls will remain disabled (and will indicate the lack of installed designs).

Loading The Demo JAM File



After selecting your target device and demonstration design, you will see the name of the "`*.jam`" file to the right of the **Download** button. Press the **Download** button to initiate loading of the the design into the attached FPGA device. Any designs that were running when **Download** is pressed will be 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. Once the selected design has been loaded and started, monitoring of the attached FPGA device will be initiated using the DCC connection.

Displaying Board Status

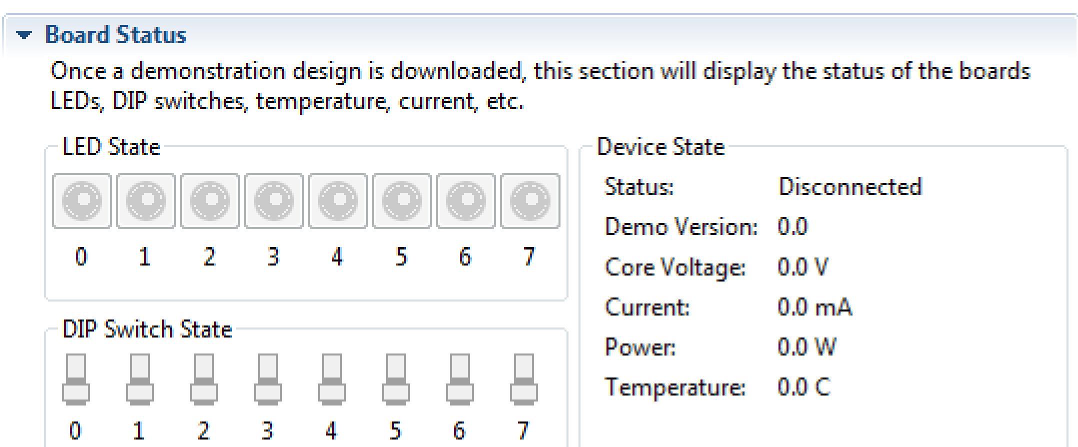
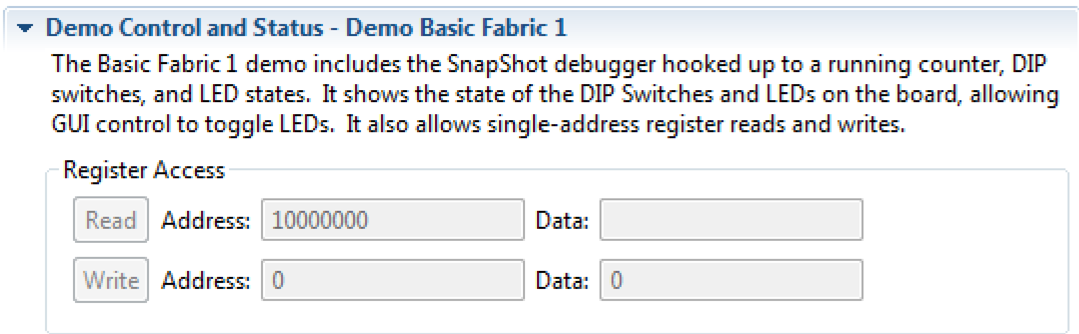


Figure 52: Example screenshot of a rudimentary demo design

After a design has been loaded and started running, ACE may monitor the status of the demonstration board LED's and DIP switches, as well as key internal conditions such as core voltage, temperature, etc. Clicking on the visualization of an LED in the HW Demo view will cause the corresponding actual LED (on the demonstration board) to toggle state.

Note that the visualized DIP switches are only used for reporting the state of the corresponding actual switch on the demonstrations board. Users cannot "flip" the physical DIP switch by clicking on its image in ACE.

Control of Running Demonstration Design



While the Snapshot Debugger has extensive facilities for collecting data samples (from a running design), it doesn't currently provide any direct mechanisms for controlling or interacting with a design. The HW Demo View (see page 189) 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 screenshot above, to read a register's value, enter its address and press the **Read** button; the current value of the specified register will appear in the **Data** field to the right. Likewise, to modify a register's value, enter its address and new value in the provided fields, and press the **Write** button.

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 "/ethernet_ref" -view "HD22i_ethernet_ref" -timestamp "1424984564" -
cp_type "hard"
```

```
set_partition_info -name "/ethernet_ref/fabric_reg" -view "ACX_MACPCS_PIPELINE_REG_521s_1s" -
timestamp "1424984564" -cp_type "locked"
set_partition_info -name "/ethernet_ref/snapshot" -view "ACX_SNAPSHOT_Z15" -timestamp
"1424984564" -cp_type "locked"
set_partition_info -name "/ethernet_ref/i_PKT" -view "packet_generator_checker_512s_6s" -
timestamp "1424984564" -cp_type "locked"
set_partition_info -name "/ethernet_ref/i_SBUS_MASTER" -view "register_control_interface" -
timestamp "1424984564" -cp_type "locked"
set_partition_info -name "/ethernet_ref/i_reset_sequencer" -view "reset_sequencer" -timestamp
"1424984564" -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. First, in the Projects View (see page 238), add the partition definition file(s) to the ACE project (see Adding Source Files (see page 355)). The new partition definition file will appear in the Projects View as a **Constraints** file and in the Options View (see page 215) in the **Design Preparation** section in the list of **Constraints Files** (should already have its checkbox selected).

Next, in the Options View (see page 215), within the **Design Preparation** section, select the checkbox labeled **Enable Incremental Compile**. Finally, in the Projects View, save the current project (see page 352). From this point forward, this Project/Implementation uses incremental compilation when running the flow.

> 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 users two main tools for checking the compilation state, timestamps, and other statistics of each Partition.

Partitions Report

The Partitions Report (see page 323), automatically generated (and opened in the GUI) during the **Run Prepare** Flow Step (see page 317), shows the user 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 231) 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's timestamp during the next pass through the Flow (see page 316). 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. Select (left-click) the partition in the Partitions View
2. Right-click anywhere in the partition's 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 will be 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. Select (left-click) the partition in the Partitions View
2. Right-click anywhere in the partition's 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 will only be re-placed and re-routed if the partition was re-compiled in the upstream synthesis tool.

Note

ACE's 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 258):

```
run -ic init
```

- Alternately:
  1. Change to the Projects Perspective
  2. In the Flow View (see page 187), enable and disable the optional Flow Steps (see page 317) 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 361) 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 245) and the Tcl command `find` 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 285) 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 249), especially when populated with search results) is an easy way to see where a partition's members are within the Floorplanner View (see page 178). When the Floorplanner's layer for **Selected Instance Flylines** is enabled, the connectivity of the Selected partition is also visible.
- The Netlist Browser View (see page 211) 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's **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 328) 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` , `get_partition_force_changed` , get_partition_info (see page 560), `get_partition_insts` , `get_partition_timestamp` , `get_partition_type` , `is_incremental_compile` , `report_partitions` , `set_partition_force_changed` , `set_partition_info`.

Additionally, the following Tcl commands were enhanced with additional options specific to incremental compilation and /or partitions: `run` , `filter` (see page 551), `find` (see page 552)

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 446) – covers how to process a single-pass incremental compile.This first tutorial must be run before running the Multiprocess Incremental Compile Tutorial
- Multiprocess Incremental Compile Tutorial (see page 474) – 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

The files needed for this tutorial are located on the Achronix FTP at:

`/Achronix/Reference_designs/Speedster22i_Incremental_Compile_RefDesign_RD009.zip`

> **Note**
>
> This is an advanced tutorial. It assumes that both Synplify Pro and ACE are in your system's search path and 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 synthesis in SynplifyPro, ACE place and route, a modification of the original RTL, and back through the flow.

The goal of the flow is to help the user 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 design's existing implementation in ACE.

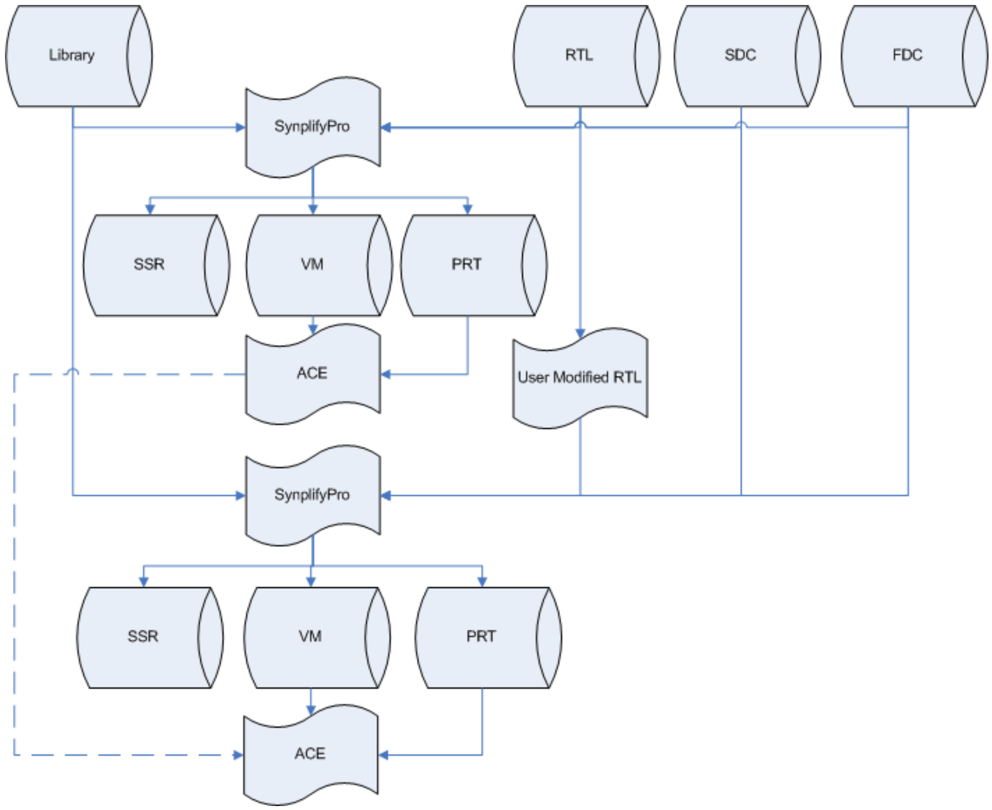


Figure 53: *Incremental Compile Flow Chart*

> **Legend**
>
> = Step that is integral to the running of this tutorial
>
> = Items that the user should check at this point in the tutorial to gain insight in to how the flow works and the feedback the tools are giving the user
>
> **Bold Text** = Text that can be found as a label to some GUI part including report table headings

Step 1: Obtain the Files

Unzip `Speedster22i_Incremental_Compile_RefDesign_RD009.zip` into a suitable work area. The archive contains the following directories:

Table 171: *Tutorial Directory Structure*

Directory	Description
`ace`	Contains all created ACE project files, reports and logs
`constraints`	Contains the SDC, PDC and FDC constraint files
`rtl rtl_V1 rtl_V2 rtl_V3 rtl_V4`	Contains the RTL files needed to create the Synplify project
`synplify`	Contains all created Synplify project, log and output files

Step 2: Set up the Synthesis Project

Start the Synplify Pro GUI. For Linux:

```
% cd <your work area>/ Speedster22i_Incremental_Compile_RefDesign_RD009
% 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>/Speedster22i_Incremental_Compile_RefDesign_RD009/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:

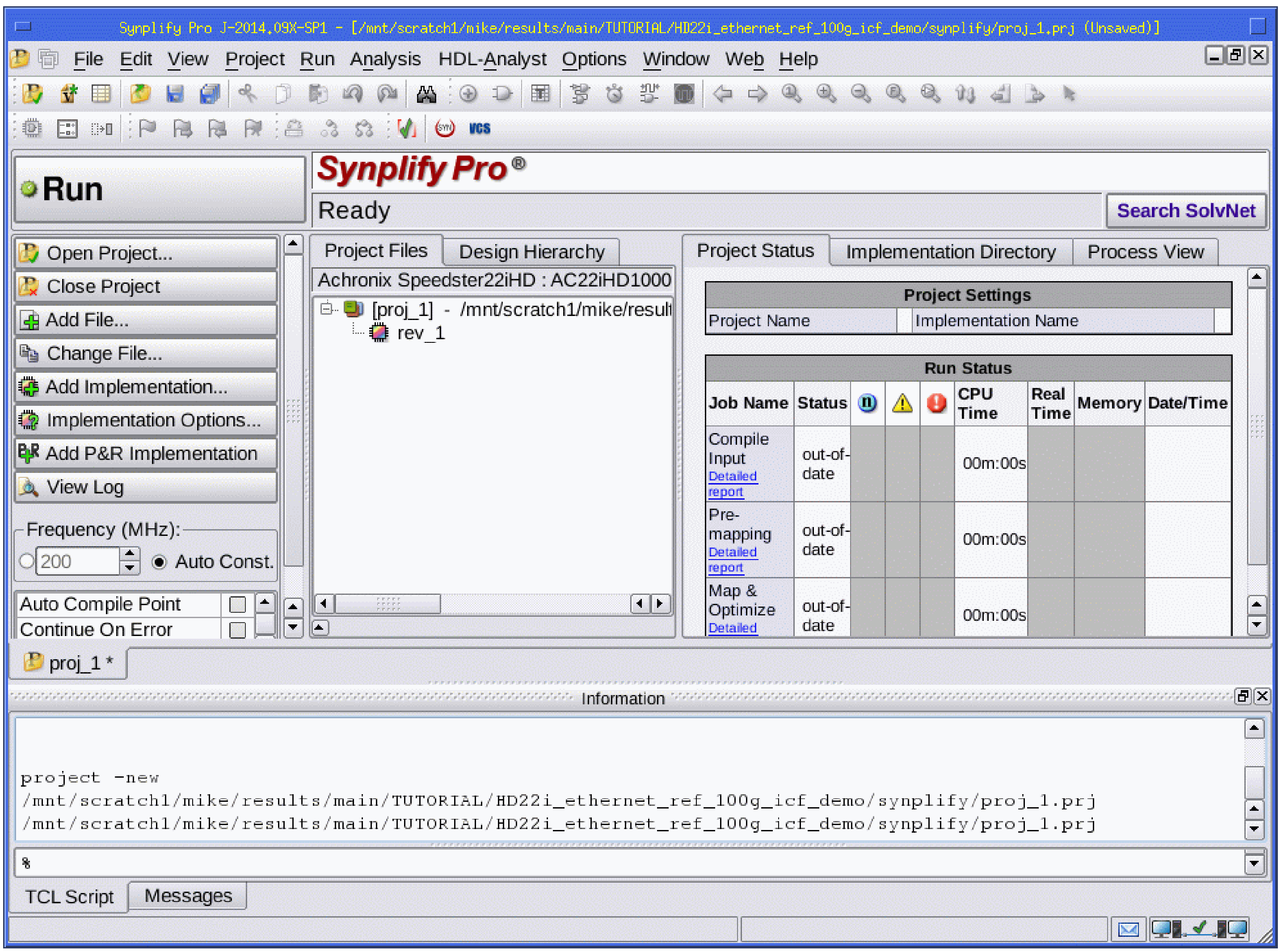


Figure 54: ***Synplify Pro Home Screen***

Select **Project** → **Add Source File** to bring up the **Add Files to Project** dialog box. Click on the blue button to navigate up to the parent directory. In the "Files of type:" drop-down box, select **All Files (*)**. Then double-click on `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**.

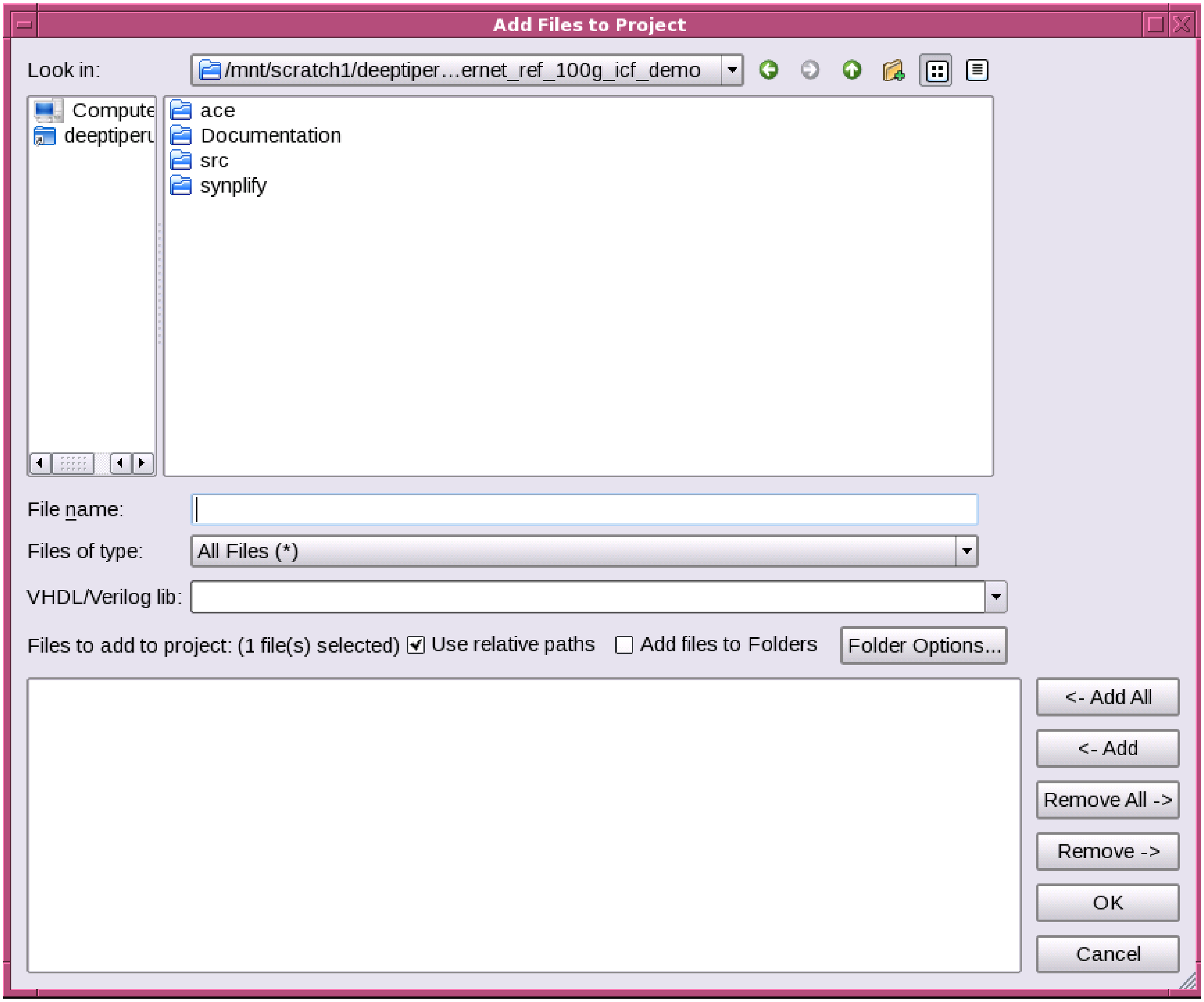


Figure 55: *"Add Files to Project" Dialog Box*

All of the files just added are now listed under the proj_1 project in the Project Files tab (click on **+** to expand each file type).

Ensure that the file `HD22i_ethernet_ref.v` (the top-level module) appears last in the list of Verilog files. If it does not, the Result Base Name implementation option under **Implementation Options** → **Implementation Results** may be set incorrectly (**Project** → **Implementation Options** → **Implementation Results** in Windows). To correct the order if needed, grab 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 Result Base Name to "HD22i_ethernet_ref_100g_icf_demo", and click **OK**. Under the Project Files tab, make note of the file named `HD22i_ethernet_ref_100g_icf_demo.fdc` that appears under the Logic Constraints (FDC) folder. This file contains the compile point definitions that enable the incremental flow.

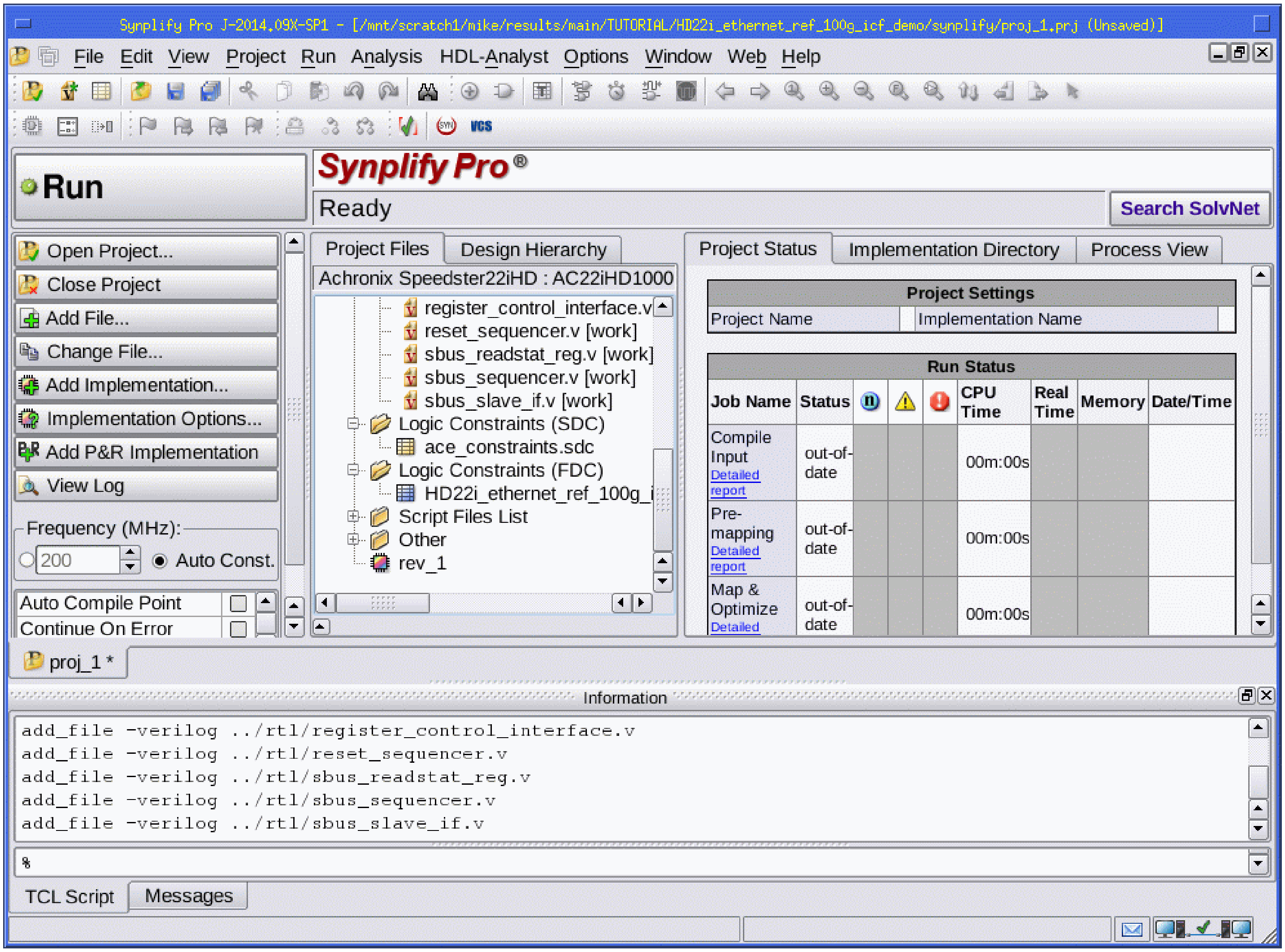


Figure 56: ***Synplify Pro Home Screen After File Additions***

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 on the green **+** and navigating to the `<ACE install location>/libraries` directory. Click **Choose**, then click **OK**.

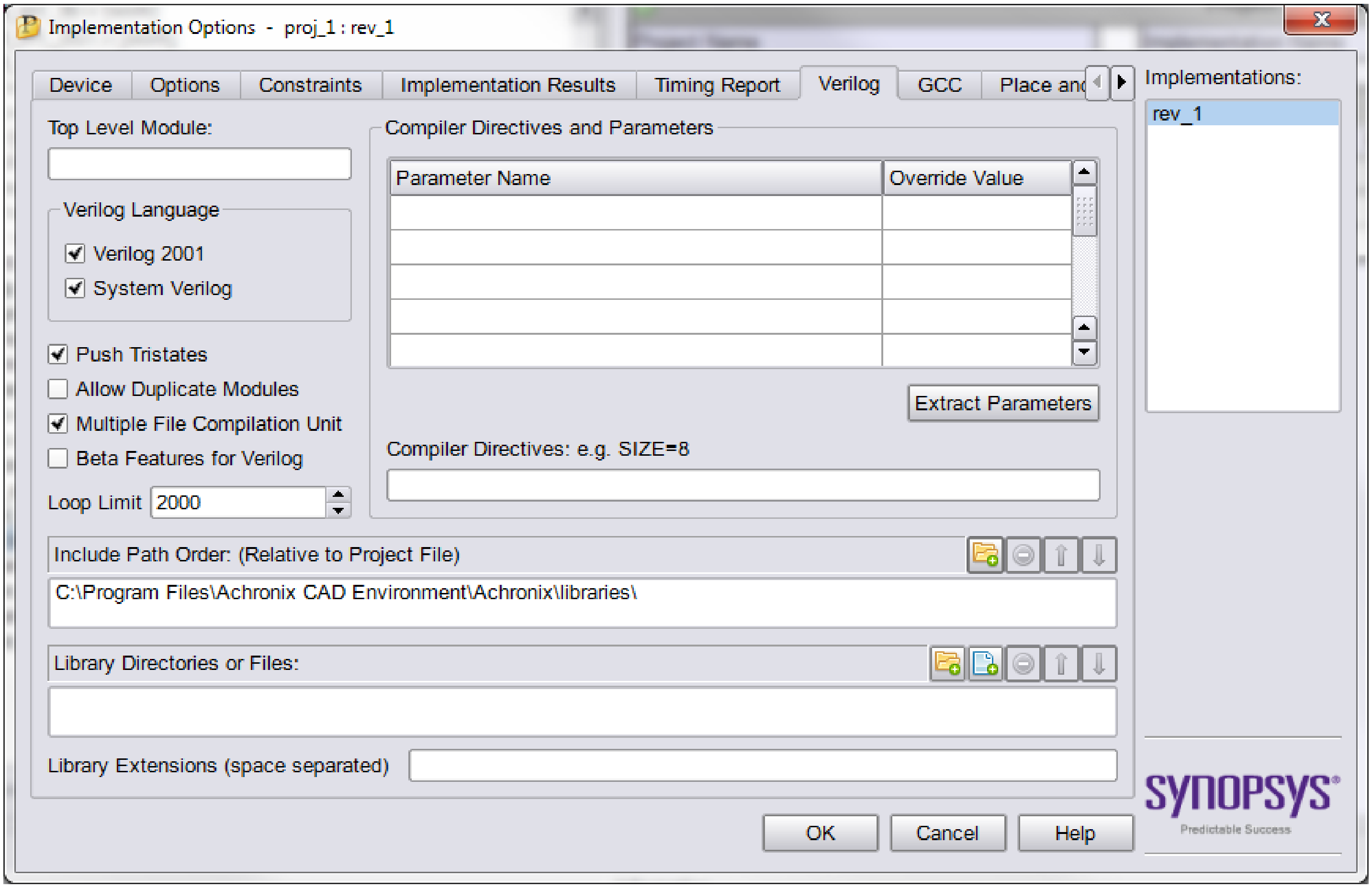


Figure 57: *Implementation Options Window*

Ensure that the Verilog file `<ACE install location>/libraries/device_models/22i_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.

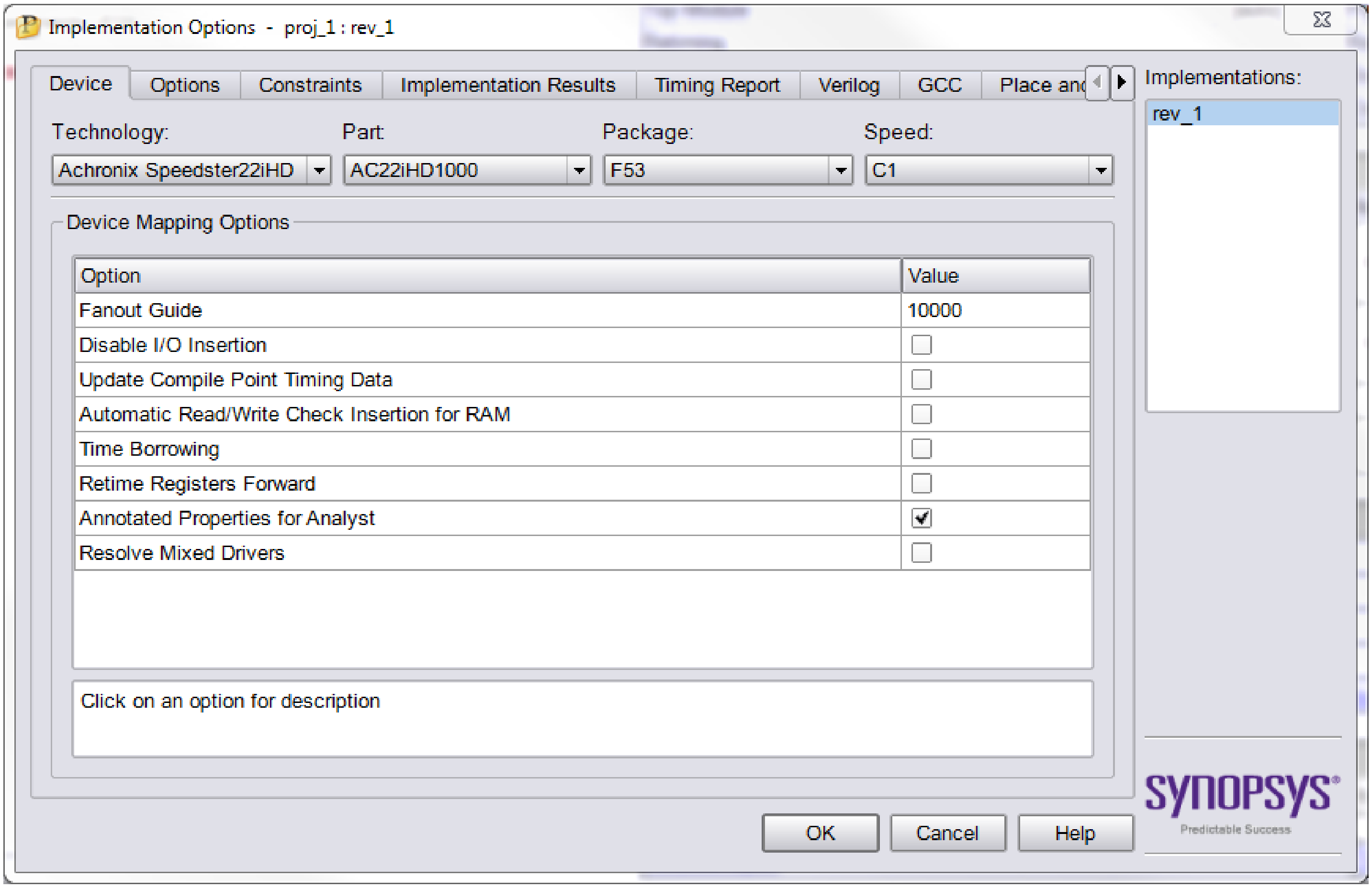


Figure 58: *Implementation Option Device Tab*

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.

Open the `HD22i_ethernet_ref_100g_icf_demo.fdc` file as a text file to view the six compile point constraints defined for this project. All six of them are of type locked, as in the example below:

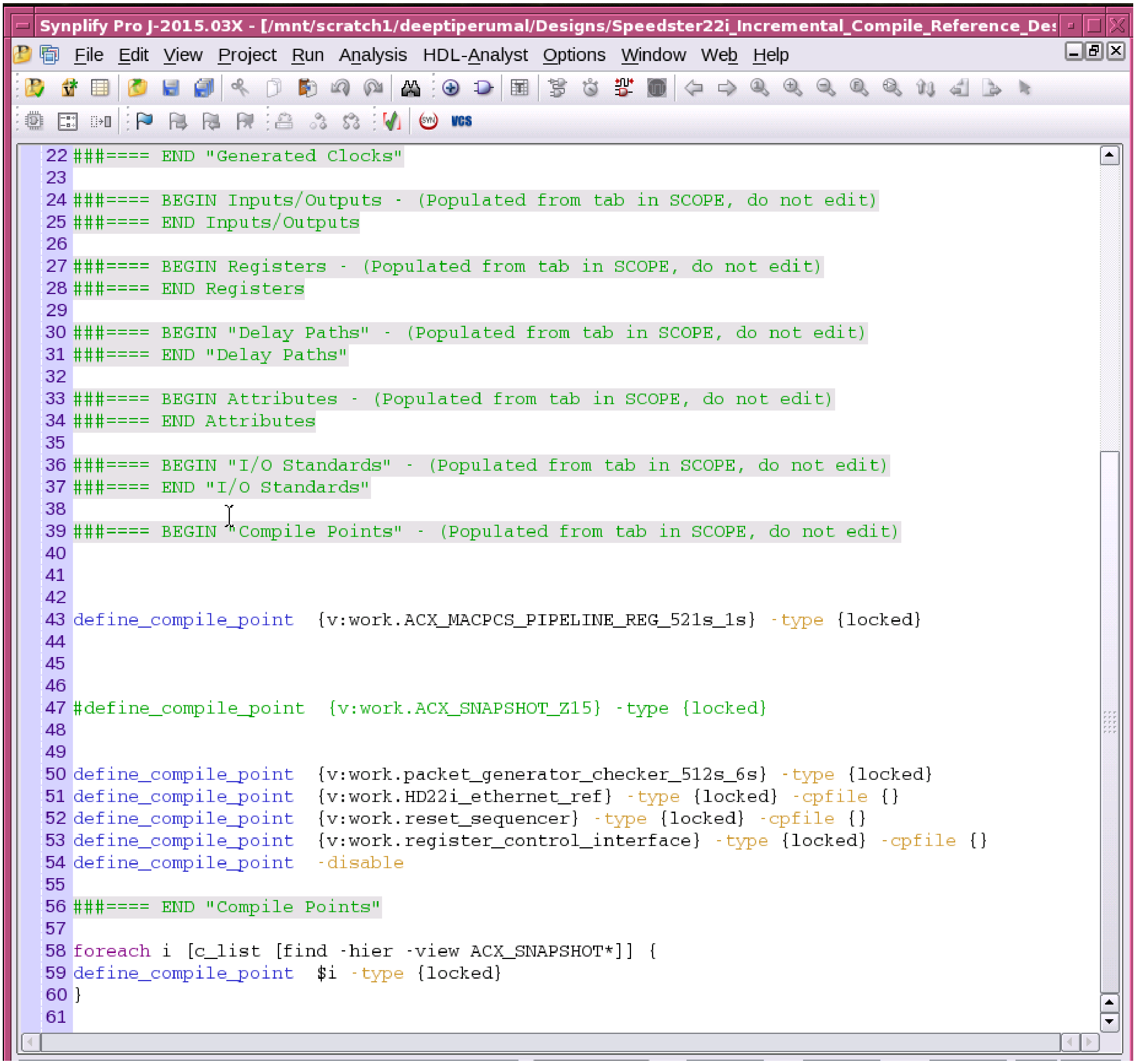


Figure 59: Contents of the *HD22i_ethernet_ref_100g_icf_demo.fdc* File

Each compile point becomes a partition in the ACE tool. If one of 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.

> **Note**
>
> The compile point view names must begin with "v:". These names are related to, but not always exactly the same as the module names in the RTL. In some cases the module names are modified with a suffix string, such as when a module has multiple instances that must be unique, or if a module is parameterized. For example,

the compile point `v:ACX_MACPCS_PIPELINE_REG_521s_1s` in this tutorial corresponds to the Verilog module named ACX_MACPCS_PIPELINE_REG. The compile point has two parameters, `.width (521)` and `.depth (1)`.

Optionally, instead of adding the `HD22i_ethernet_ref_100g_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 `HD22i_ethernet_ref_100g_icf_demo.fdc` file in the SCOPE tool, double-click on 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.

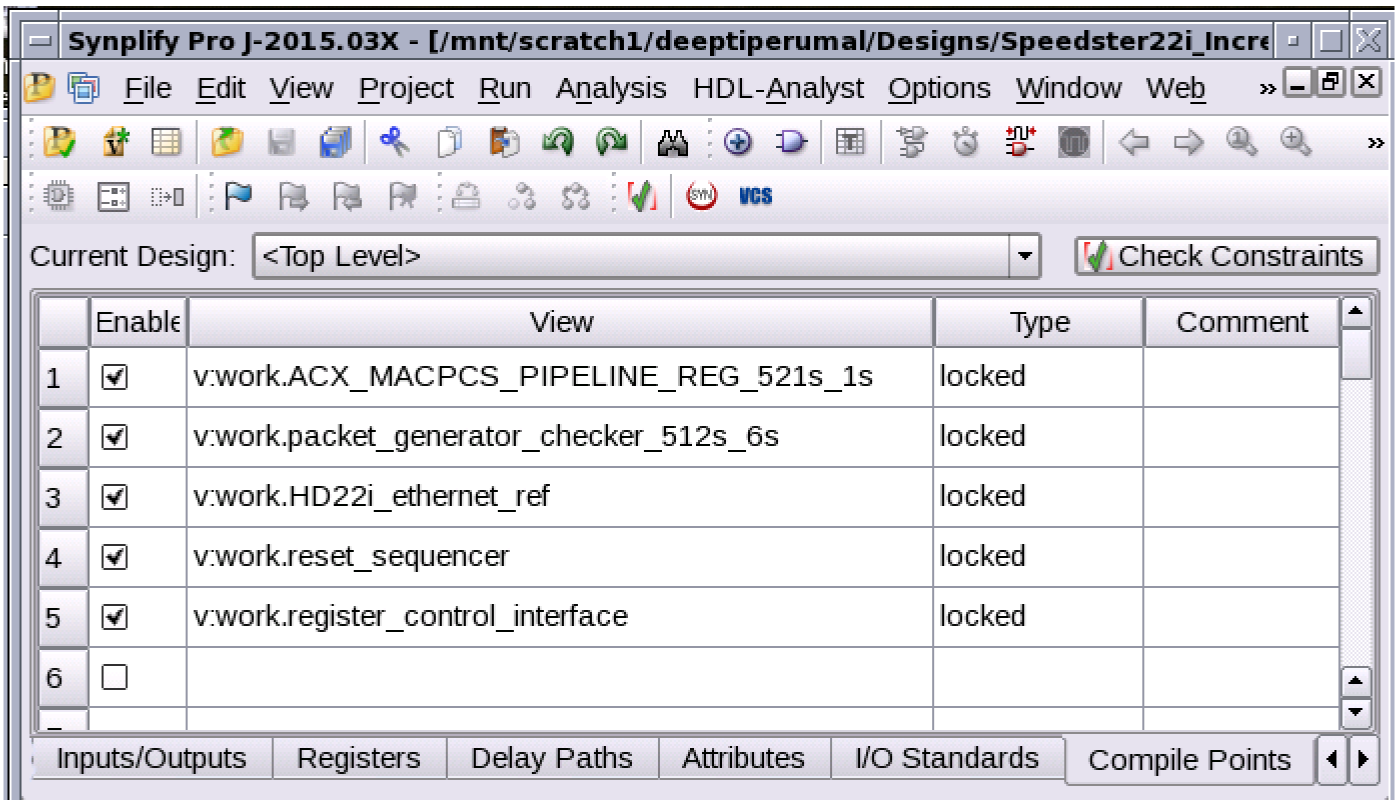


Figure 60: *"Compile Points" Tab Within the Synplify Pro SCOPE View*

Note

When opening the Compile Points tab of the SCOPE View, it shows only five compile points. The compile point for the ACX_SNAPSHOT view must be created in the `.fdc` file with a Tcl `foreach` loop and `find` command because the view name suffix for the ACX_SNAPSHOT macro has been observed to change over time. The Compile Points tab is unable to display that constraint; however, it is listed in the Tcl View tab of the SCOPE tool.

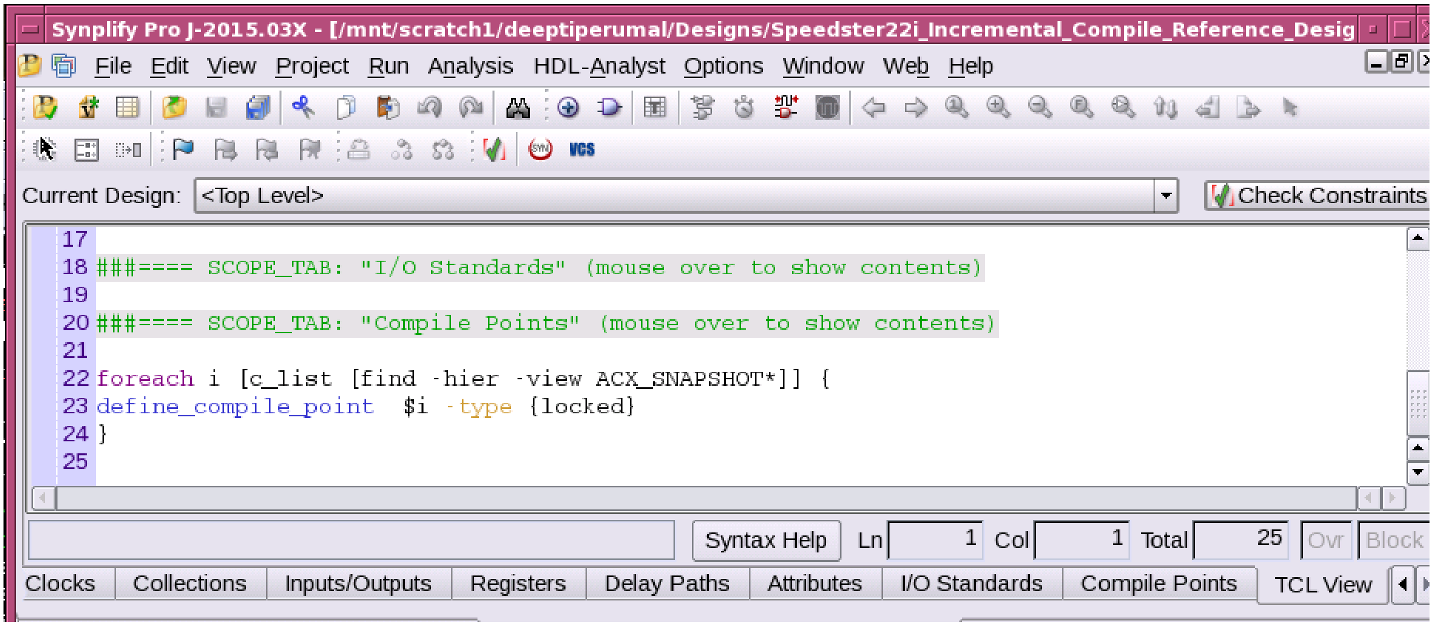


Figure 61: *"TCL View" Tab Within the Synplify Pro SCOPE View*

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.

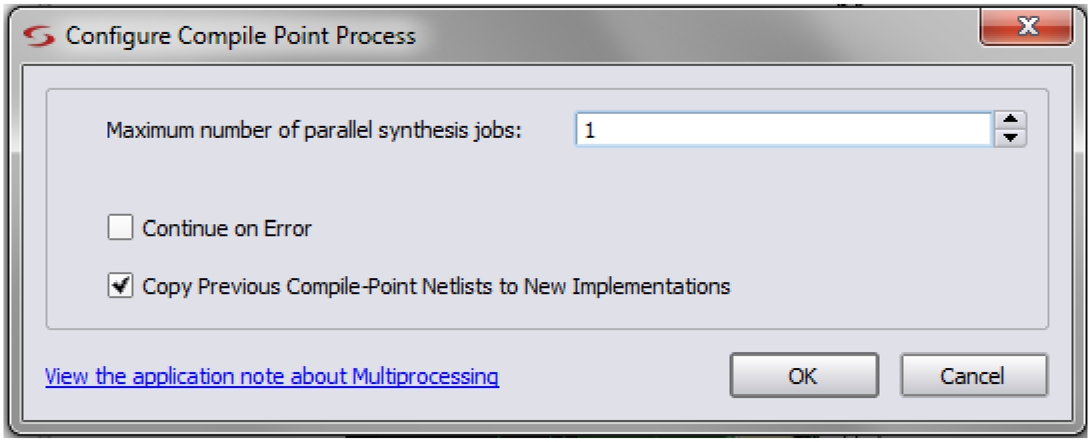


Figure 62: ***Configure Compile Point Process Dialog Box***

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 `Speedster22i_Incremental_Compile_Reference_Design_RD009/synplify/rev_1/HD22i_ethernet_ref.srr` and search for the section titled "Summary of Compile Points".

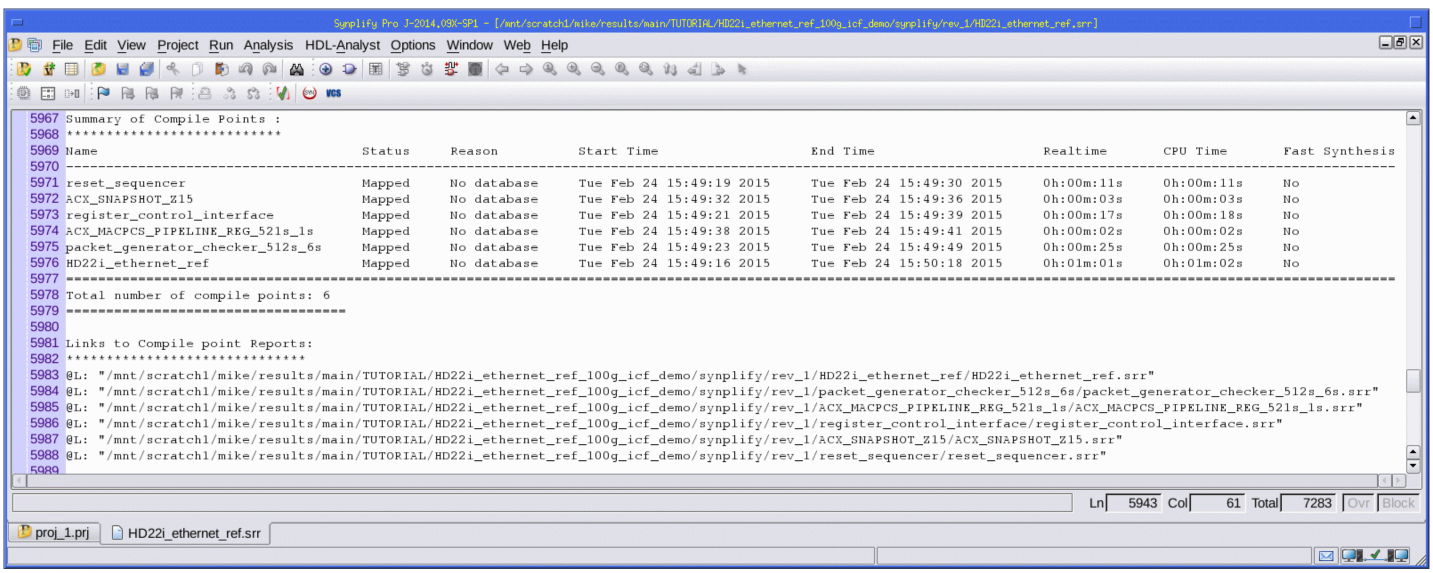


Figure 63: ***Synplify-Pro Log File Showing "Summary of Compile Points" Section***

✔ 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.reset_sequencer(verilog)
@N:MF104 : | Found compile point of type locked on View view:work.register_control_interface
(verilog)
@N:MF104 : | Found compile point of type locked on View view:work.
packet_generator_checker_512s_6s(verilog)
@W:BN539 : | syn_hier=hard on compile point ACX_SNAPSHOT_Z16 not supported. Removing
syn_hier=hard.
@N:MF104 : | Found compile point of type locked on View view:work.ACX_SNAPSHOT_Z15(verilog)
@N:MF104 : | Found compile point of type locked on View view:work.
ACX_MACPCS_PIPELINE_REG_521s_1s(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 `HD22i_ethernet_ref_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.

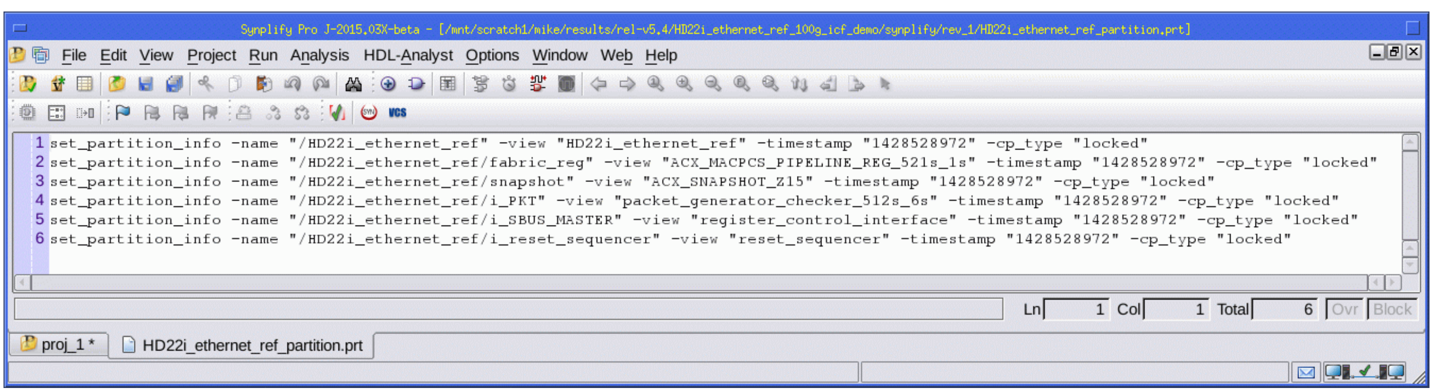


Figure 64: Contents of the `HD22i_ethernet_ref_partition.prt` File

Technology View

Click on 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 on one to select it. The selected instance is highlighted with a red boundary in the Tech popup view.

Use the right mouse to push into that level of the hierarchy. The schematic will then update. 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.

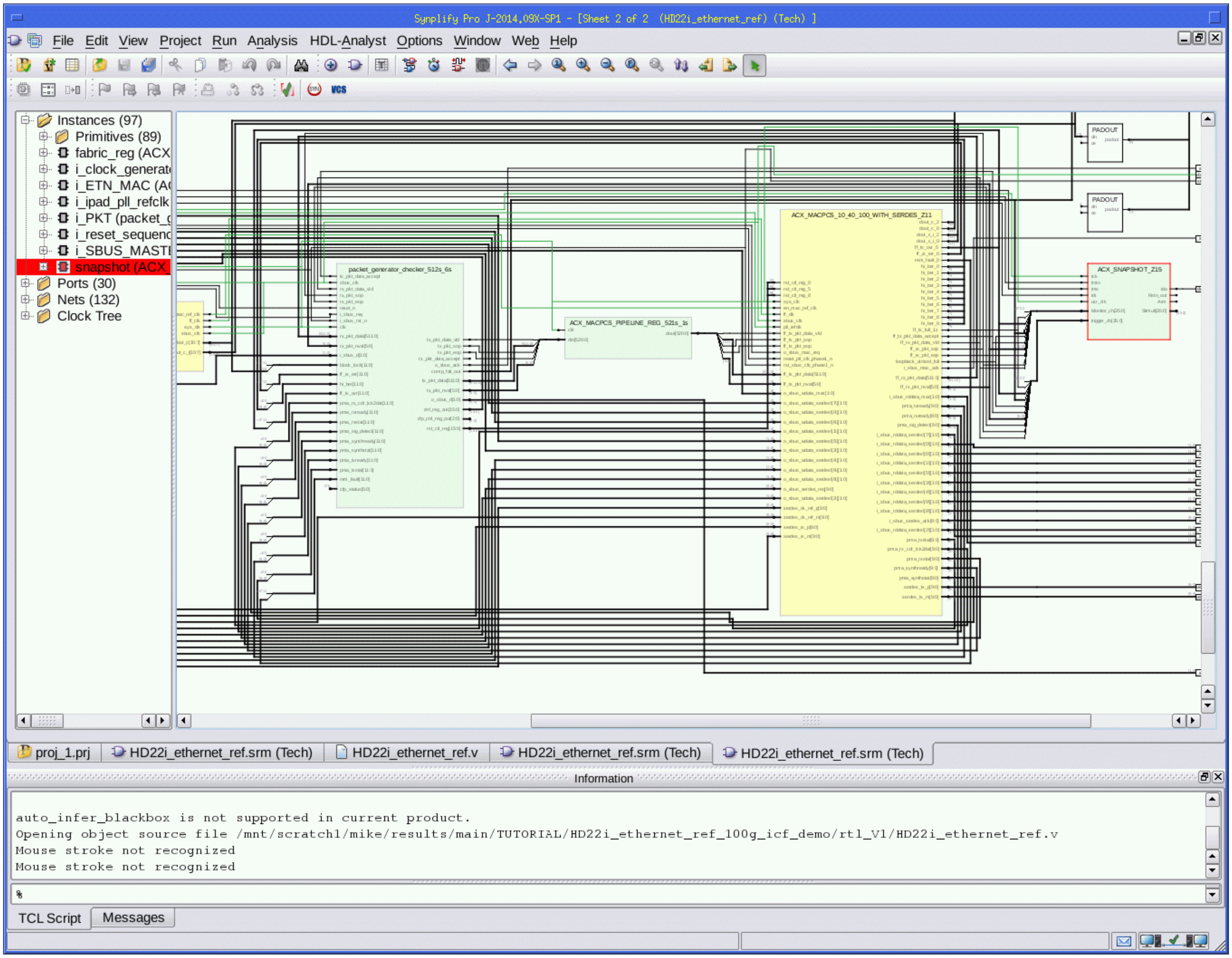


Figure 65: *Synplify Pro "Technology View"*

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>/Speedster22i_Incremental_Compile_RefDesign_RD009/ace
% ace
```

Under Windows, double click on 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 `Speedster22i_Incremental_Compile_RefDesign_RD009/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:

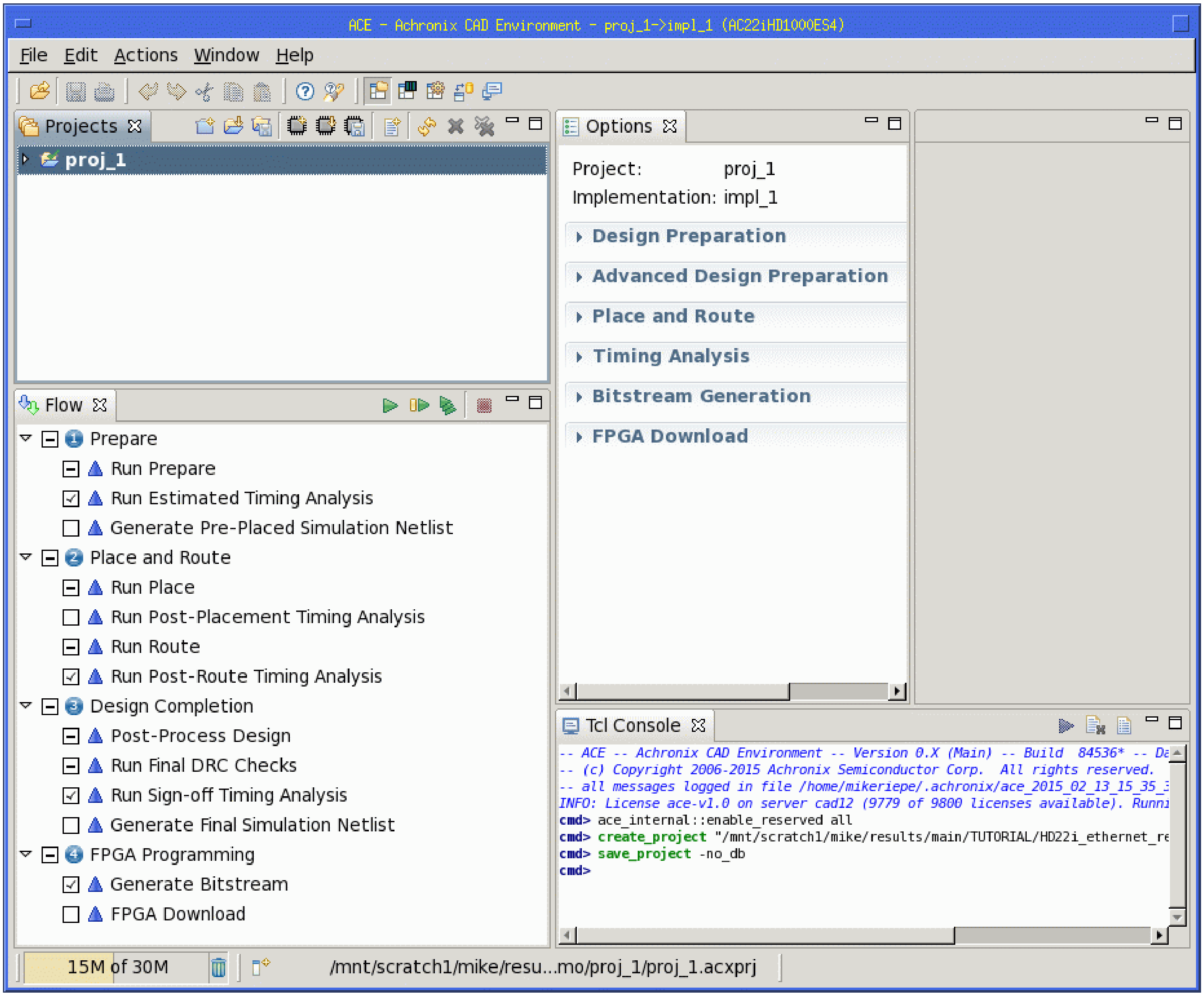


Figure 66: ***ACE Home Screen***

Select **File** → **Add Project Source Files…**, then click on **Speedster22i_Incremental_Compile_RefDesign_RD009** in the pathname bar to locate the source files and open the following dialog box:

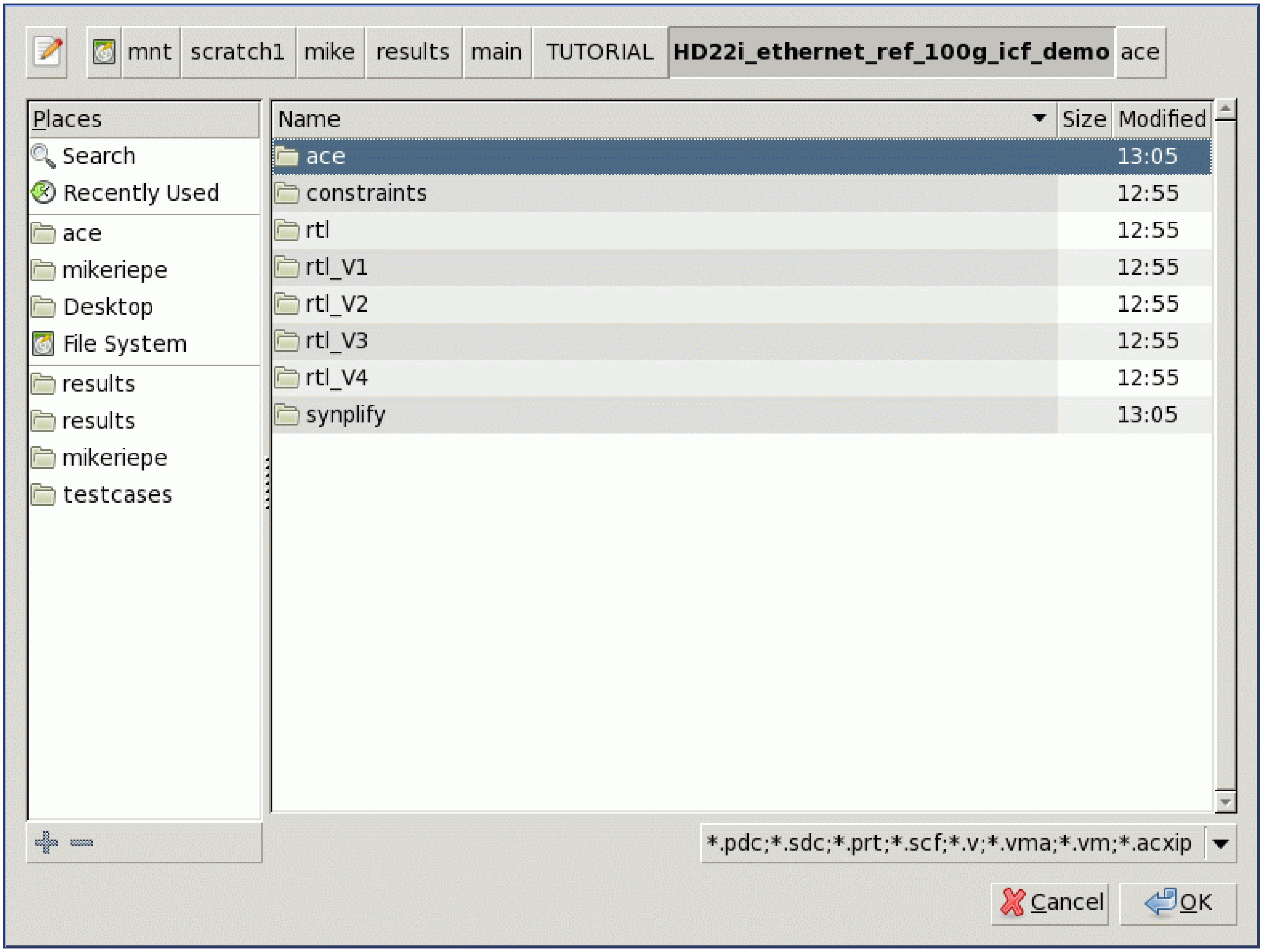


Figure 67: *"Design Source Files" Dialog Box*

Navigate to the constraints directory, shift-click to select all three files, 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 `HD22i_ethernet_ref.vm` and `HD22i_ethernet_ref_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 on the triangle next to Constraints to list out the constraint files, etc.

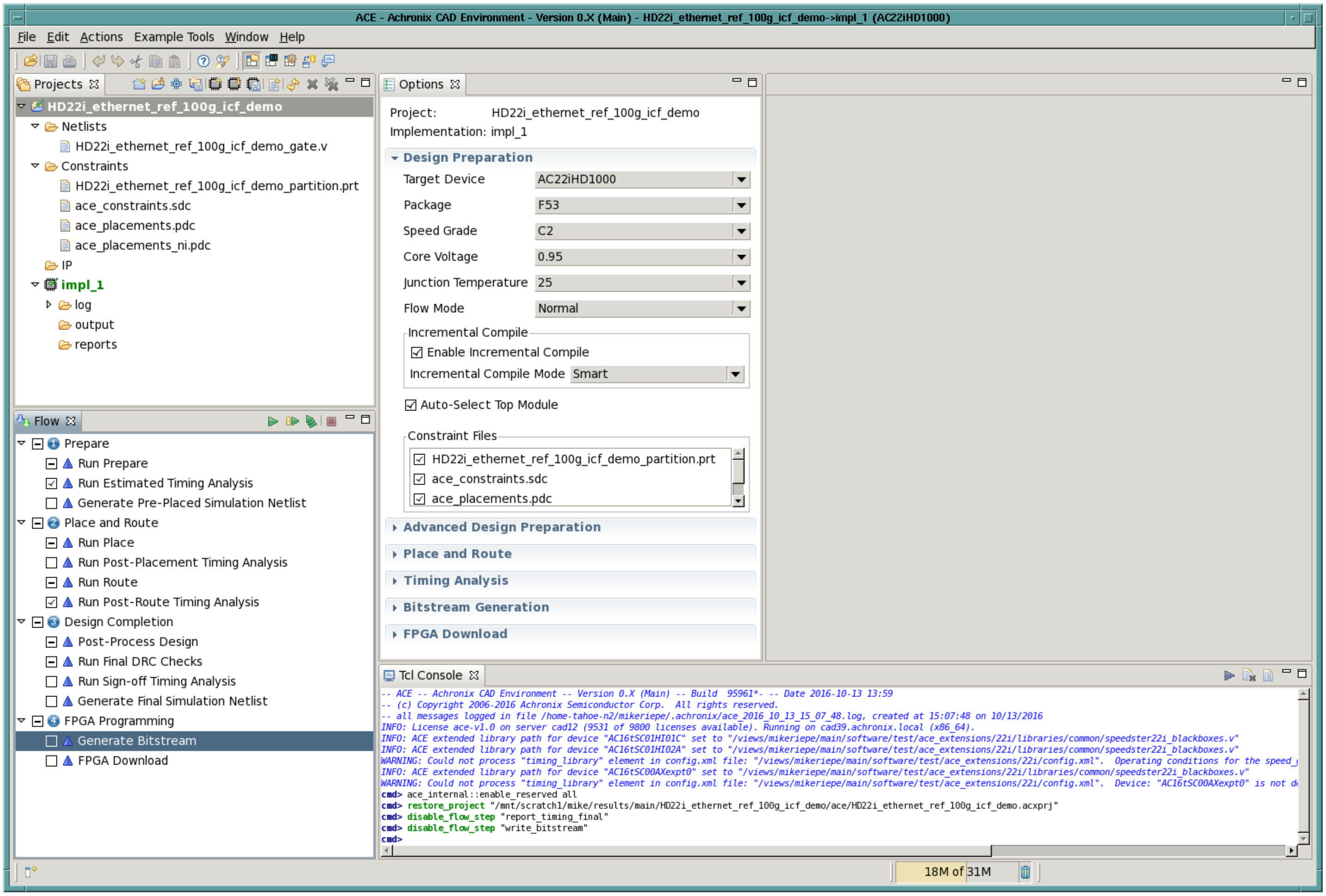


Figure 68: *ACE "Projects" Tab*

✅ Recall from Step 4 (see page) that the `HD22i_ethernet_ref_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

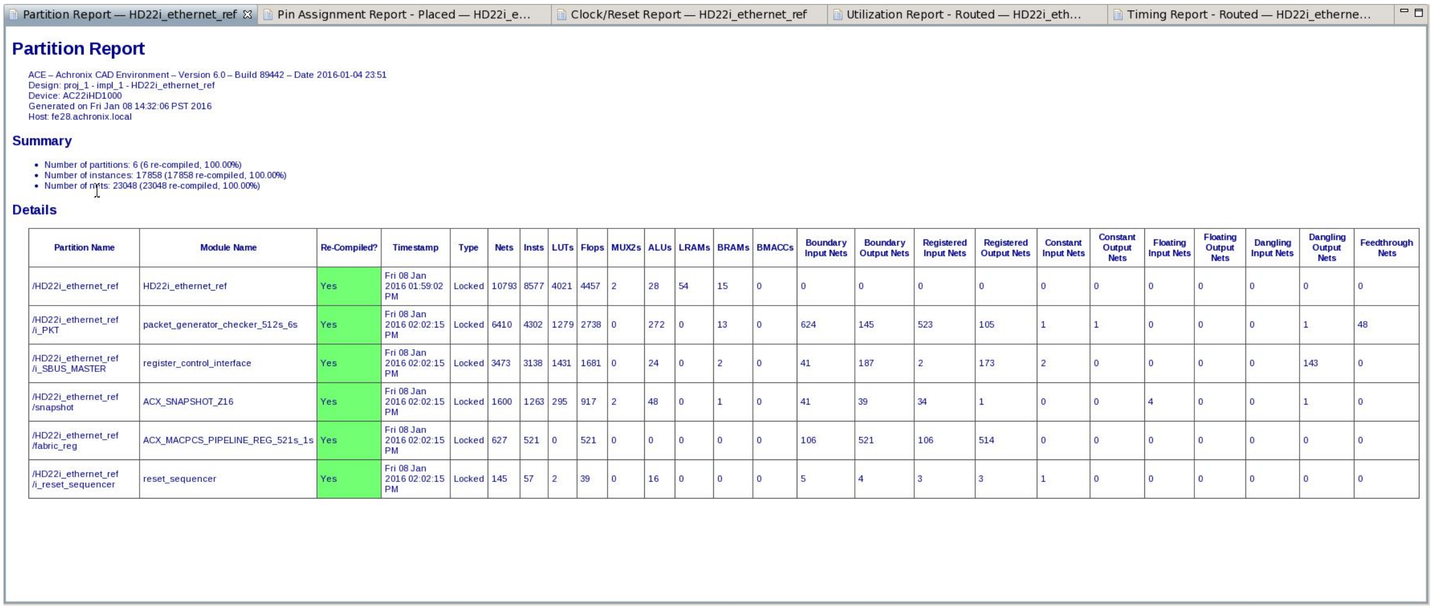
➕ In the Flow tab, uncheck the **Run Sign-off Timing Analysis** and **Generate Bitstream** flow step checkboxes to save some runtime. Then click the green triangle  in the upper-right corner of the Flow View to run the Prepare, Placement, and Routing flow. When the ACE flow completes, a green check mark 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 6.0 – Build 89442 – Date 2016-01-04 23:51
Design: proj_1 - impl_1 - HD22i_ethernet_ref
Device: AC22iHD1000
Generated on Fri Jan 08 14:32:06 PST 2016
Host: fe28.achronix.local

Summary

- Number of partitions: 6 (6 re-compiled, 100.00%)
- Number of instances: 17858 (17858 re-compiled, 100.00%)
- Number of nets: 23048 (23048 re-compiled, 100.00%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX2s	ALUs	LRAMs	BRAMs	BMACCs	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
/HD22i_ethernet_ref	HD22i_ethernet_ref	Yes	Fri 08 Jan 2016 01:59:02 PM	Locked	10793	8577	4021	4457	2	28	54	15	0	0	0	0	0	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_PKT	packet_generator_checker_512s_6s	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	6410	4302	1279	2738	0	272	0	13	0	624	145	523	105	1	1	0	0	0	1	48
/HD22i_ethernet_ref /i_SBUS_MASTER	register_control_interface	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	3473	3138	1431	1681	0	24	0	2	0	41	187	2	173	2	0	0	0	0	143	0
/HD22i_ethernet_ref /snapshot	ACX_SNAPSHOT_Z16	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	1600	1263	295	917	2	48	0	1	0	41	39	34	1	0	0	4	0	0	1	0
/HD22i_ethernet_ref /fabric_reg	ACX_MACPCS_PIPELINE_REG_521s_1s	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	627	521	0	521	0	0	0	0	0	106	521	106	514	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_reset_sequencer	reset_sequencer	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	145	57	2	39	0	16	0	0	0	5	4	3	3	1	0	0	0	0	0	0

Figure 69: ***ACE Partition Report After the 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 will be 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 will be 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's flyout palette, under the Layers section, turn on visibility for Instances, but turn off visibility for Sites, Clock Routes, and Non-clock Routes.

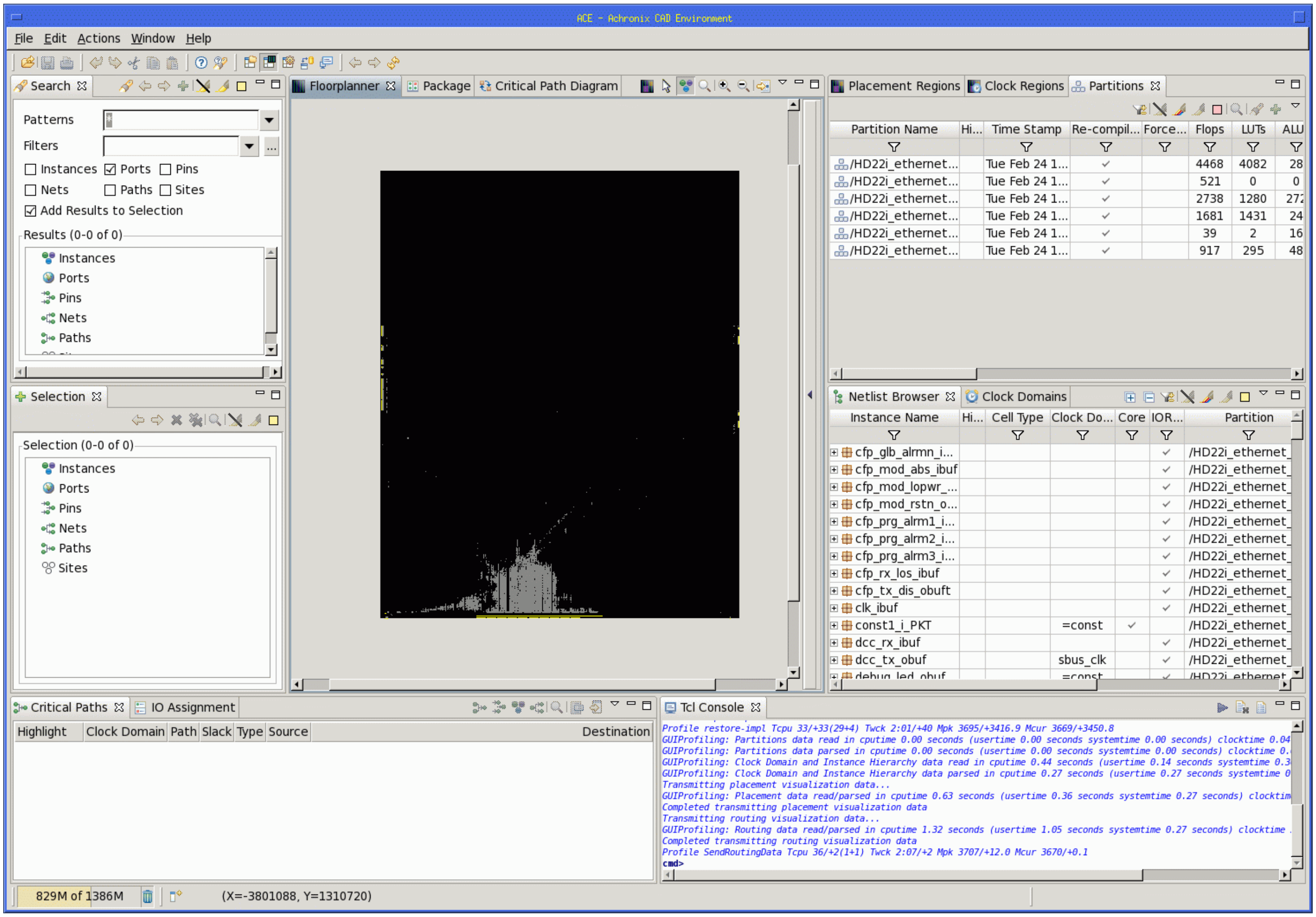


Figure 70: *ACE Floorplanner Perspective After the First Incremental Compile Iteration*

➕ First switch to the Partitions tab to view an active 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 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 number of instances of each type includes only those instances directly owned by the given partition.

The column named Force Recompile on Next Run provides a mechanism to override the partition's 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, 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.

➕ The column named Highlight Color displays a box with the partition's highlight color. It should currently be empty. Right-click on the row for a partition and select Highlight Partition to highlight the partition's instances in the Floorplanner view with the current highlight color for the Partition tab. In the toolbar above the table, click on the Choose Highlight

Color tool to change the highlight color to be used in the next Highlight command. Right-click on the partition row and select **Un-Highlight Partition** to disable highlighting of the partition's 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:

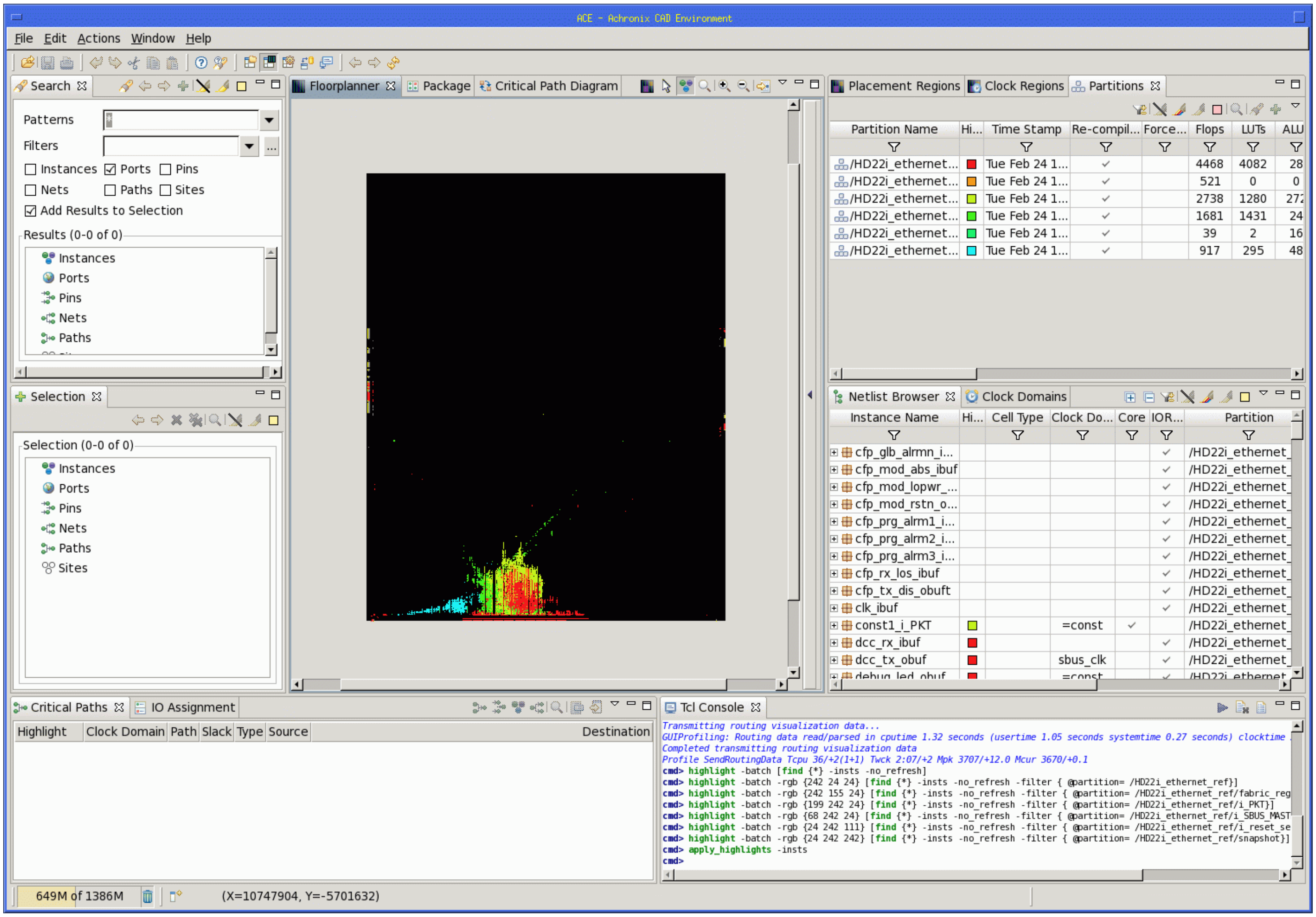


Figure 71: *Highlighting Partitions in ACE After the First Incremental Compile Iteration*

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 to the instances of the selected partition, search for the instances of the selected partition (this generates the appropriate TCL find command and prints the resulting list of instances to the console), and add the instances of the selected partition to the selection set.

To enable the top row of the Partition table contain filter icons which can be used to control visibility of the partition rows, click on the **Toggle Filter Row Visibility** button . Clicking on this icon allows a visibility filter based on the values in that column to be specified. For example, clicking on the filter icon above the LUTs column allows viewing only partitions with, say, greater than 3,000 LUTs. This filter can be useful if there are a large number of partitions.

Tip

When using the Placement Region Tool in the Floorplanner view to create a new placement region, drag-and-drop a row from the Partitions view onto the newly created region. This action generates the correct add_region_find_insts command to add all instances in that partition to the placement region.

Netlist Browser

The Netlist Browser tab also contains several features dedicated to the Incremental Compile Flow. There is a new column named Partition listing 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.

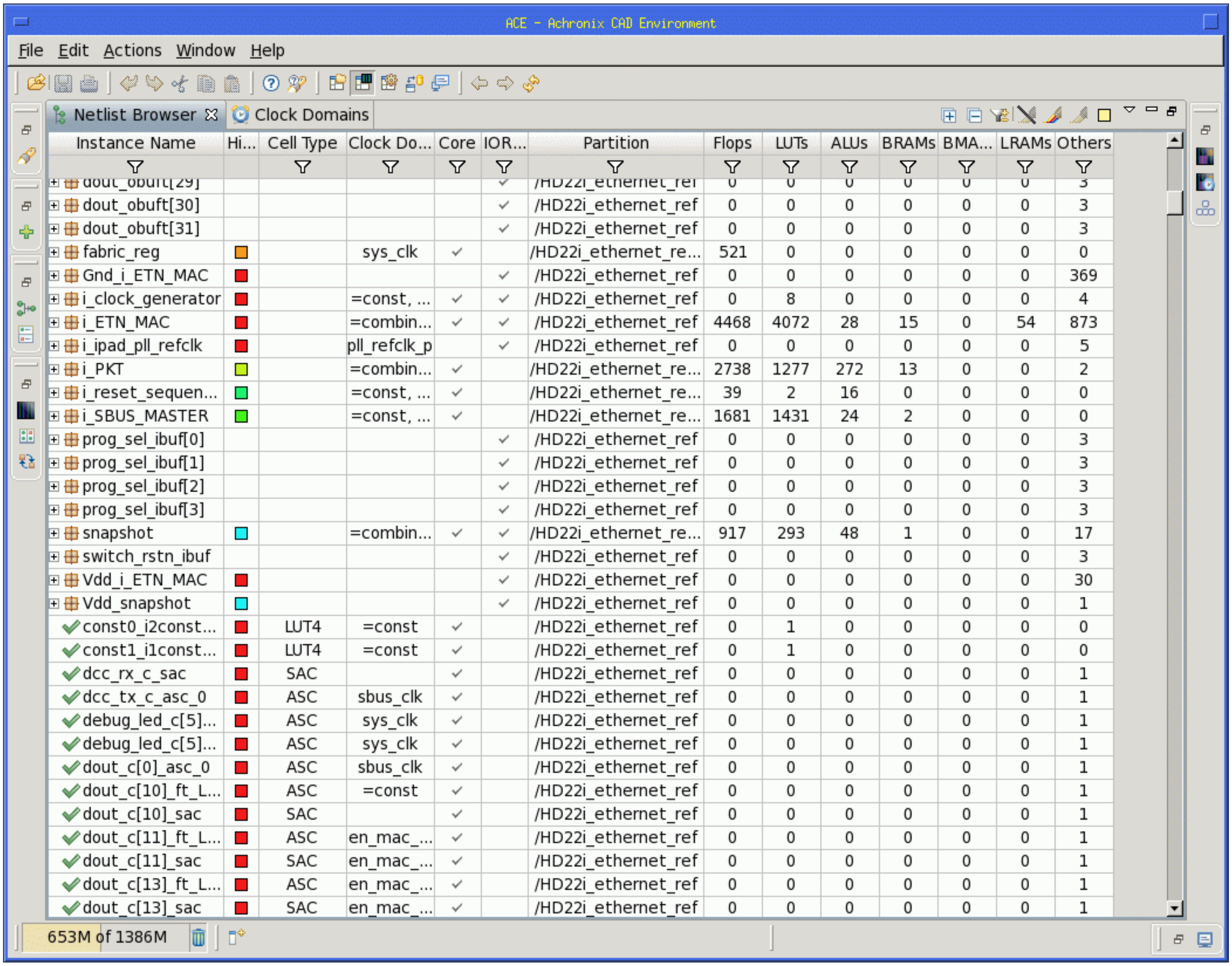


Figure 72: *ACE Netlist Browser*

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 will recognize that inputs to the ACE flows have changed, and will run 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>/ Speedster22i_Incremental_Compile_Reference_Design_RD009
% synplify_pro
```

Under Windows, double-click on the Synplify Pro icon.

This step simulates an RTL change by replacing the source file `rtl/HD22i_ethernet_ref.v` with the version in the directory `rtl_V1`. To do this in Synplify Pro, select the file to be changed by clicking on its name in the Project Files tab, in this case `HD22i_ethernet_ref.v`, and then click **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 on the file `HD22i_ethernet_ref.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/HD22i_ethernet_ref.v rtl
```

> **Note**
>
> This design uses the Snapshot™ debugger macro — one of the nets being monitored has been removed from `rtl_V1`.

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 will be 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 `Speedster22i_Incremental_Compile_Reference_Design_RD009/synplify/rev_1/HD22i_ethernet_ref.srr` and search for the section titled "Summary of Compile Points". All of the partitions have a status of *unchanged* except for the top-level partition `HD22i_ethernet_ref`, which has a status of *remapped* and a reason of *design changed*. The timestamp of the remapped partition has also been advanced.

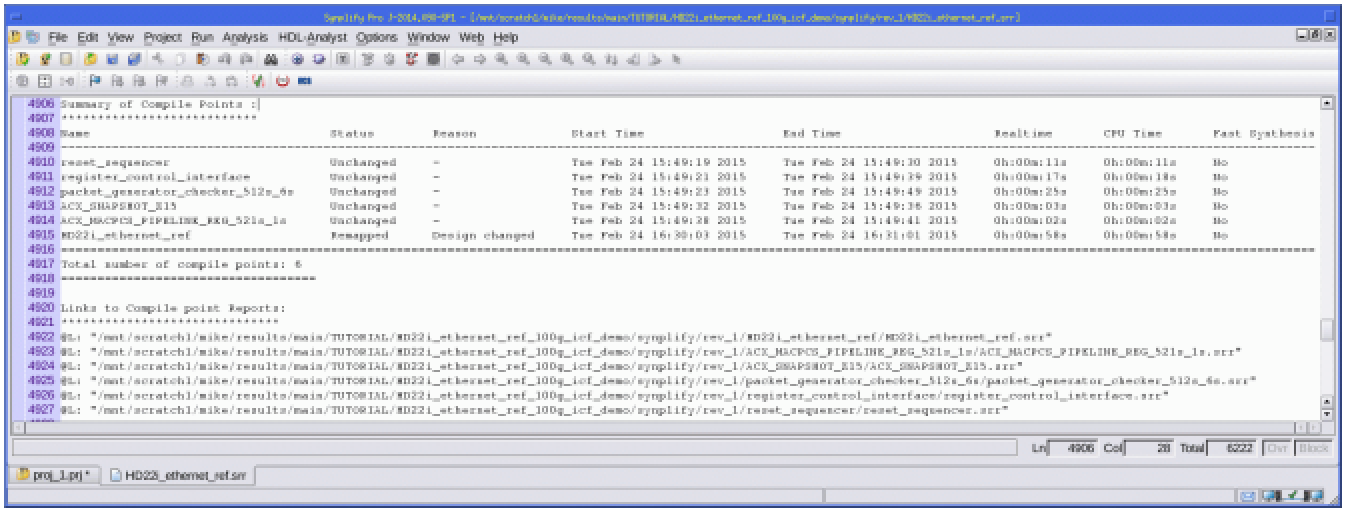


Figure 73: ***Synplify Pro Log File Showing Changed Compile Points***

ACE Partitioning Constraints File (rtl_V1)

Also re-open file `Speedster22i_Incremental_Compile_RefDesign_RD009/synplify/rev_1/HD22i_ethernet_ref_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.

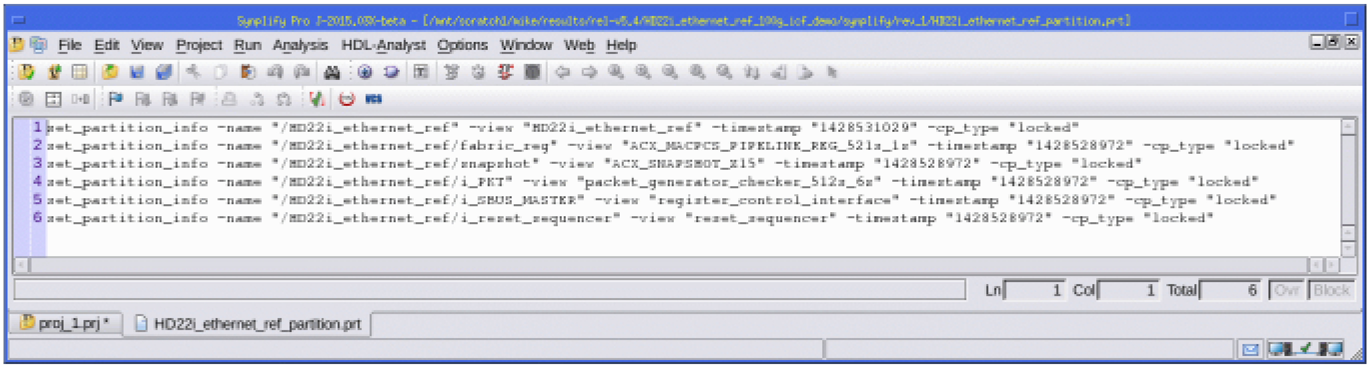


Figure 74: ***ACE Partitioning Constraints File: HD22i_ethernet_ref_partition.prt***

Use **File** → **Close** to close Synplify Pro, and click on "**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.

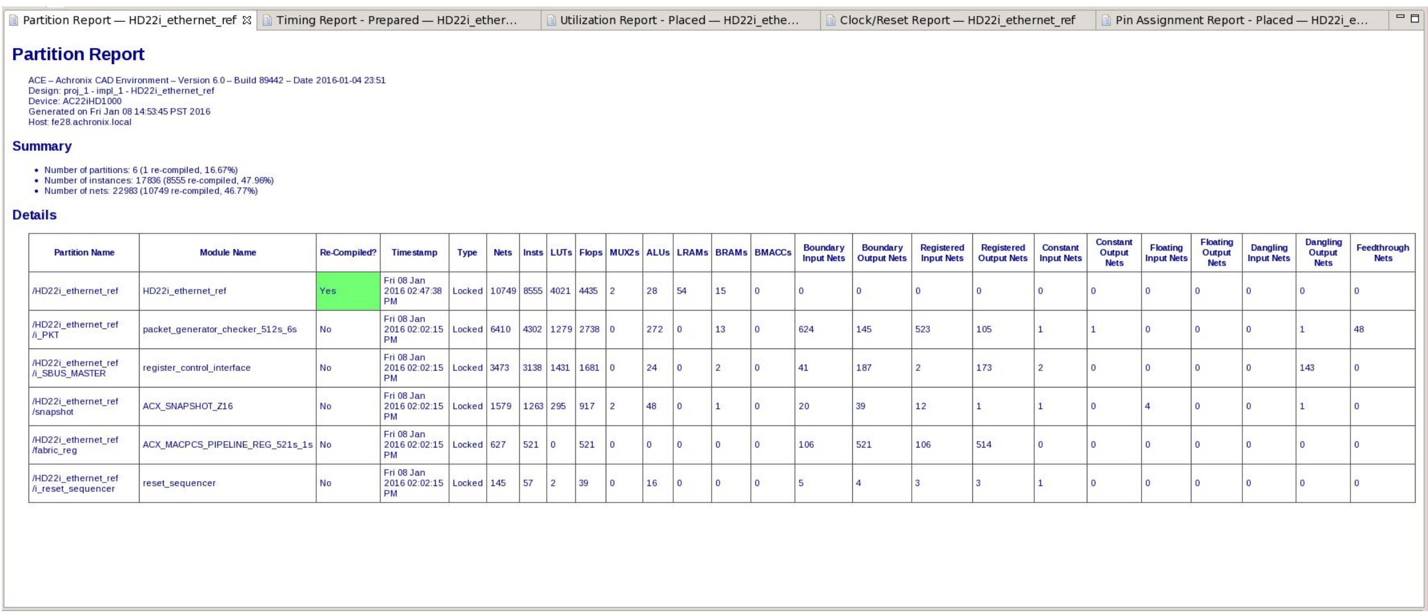
> **Note**
>
> The runtime of the flow is also significantly reduced over the first non-incremental compile.

In this second pass, ACE reads the new `HD22i_ethernet_ref.vm` netlist file and the new `HD22i_ethernet_ref_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)

In the Projects perspective select the Partition Report tab and maximize it using the **Maximize** button. In the summary section, only 1 of the 6 partitions (16.67%) was recompiled by ACE, resulting in 48.24% of the instances being re-placed and 49.67% of the nets being rerouted. Also note in the Details section that it was the top-level partition that was recompiled, and its new timestamp is displayed. The counts of instances and nets in that partition have changed by a small amount.


Partition Report — HD22i_ethernet_ref | Timing Report - Prepared — HD22i_ether... | Utilization Report - Placed — HD22i_ethe... | Clock/Reset Report — HD22i_ethernet_ref | Pin Assignment Report - Placed — HD22i_e...

Partition Report

ACE – Achronix CAD Environment – Version 6.0 – Build 89442 – Date 2016-01-04 23:51
Design: proj_1 - impl_1 - HD22i_ethernet_ref
Device: AC22iHD1000
Generated on Fri Jan 08 14:53:45 PST 2016
Host: fe28.achronix.local

Summary

- Number of partitions: 6 (1 re-compiled, 16.67%)
- Number of instances: 17836 (8555 re-compiled, 47.96%)
- Number of nets: 22983 (10749 re-compiled, 46.77%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX2s	ALUs	LRAMs	BRAMs	BMACCs	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
/HD22i_ethernet_ref	HD22i_ethernet_ref	Yes	Fri 08 Jan 2016 02:47:38 PM	Locked	10749	8555	4021	4435	2	28	54	15	0	0	0	0	0	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_PKT	packet_generator_checker_512s_6s	No	Fri 08 Jan 2016 02:02:15 PM	Locked	6410	4302	1279	2738	0	272	0	13	0	624	145	523	105	1	1	0	0	0	1	48
/HD22i_ethernet_ref /i_SBUS_MASTER	register_control_interface	No	Fri 08 Jan 2016 02:02:15 PM	Locked	3473	3138	1431	1681	0	24	0	2	0	41	187	2	173	2	0	0	0	0	143	0
/HD22i_ethernet_ref /snapshot	ACX_SNAPSHOT_Z16	No	Fri 08 Jan 2016 02:02:15 PM	Locked	1579	1263	295	917	2	48	0	1	0	20	39	12	1	1	0	4	0	0	1	0
/HD22i_ethernet_ref /fabric_reg	ACX_MACPCS_PIPELINE_REG_521s_1s	No	Fri 08 Jan 2016 02:02:15 PM	Locked	627	521	0	521	0	0	0	0	0	106	521	106	514	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_reset_sequencer	reset_sequencer	No	Fri 08 Jan 2016 02:02:15 PM	Locked	145	57	2	39	0	16	0	0	0	5	4	3	3	1	0	0	0	0	0	0

Figure 75: ***ACE Partition Report After the rtl_V1 Incremental Compile Iteration***

Floorplanner View (rtl_V1)

Switch to the Floorplanner perspective and turn off visibility for placement region region_0. In the Floorplanner view, compared to the figure above (see page 464), 48% of the instances are now drawn with a dark-yellow fill color, which indicates their locations are locked.

If not, select **Window → Preferences → Floorplanner View Colors** and ensure that the **Instances → Show Locked Color on Instances with Locked Placement** box is checked, and that the **Locked Instances View Color** is set to any desired color to represent locked instances and note that color changes during incremental iterations (click **Restore Defaults** if necessary). This background color is similar to the light-yellow background color used to call out instances with fixed placement. The Locked Instances View Color calls out instances that are locked in place because they are in a partition that was not recompiled. Only instances with the light-grey background color required replacement.

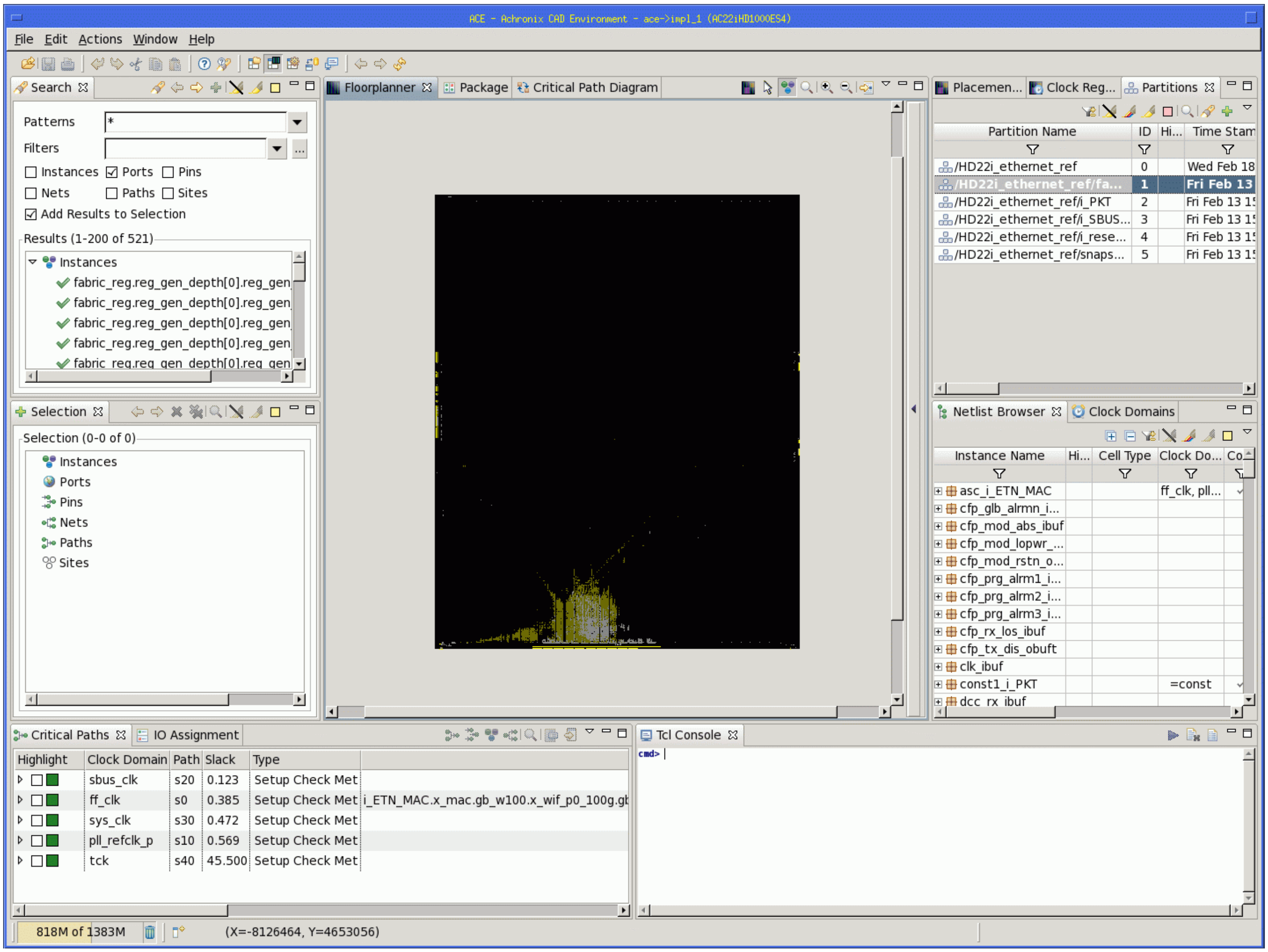


Figure 76: *ACE Floorplanner Perspective After the rtl_V1 Incremental Compile Iteration*

To get a feel for the amount of re-routing that was required, turn on the visibility of viewing selected flylines by checking the **Selected Instance Flylines** box in the Floorplanner view. Then in the Partitions view, select the partitions that were recompiled, and click **Add Instances to Selection** in the Partitions View toolbar. Blue flylines are drawn for all rerouted nets (or rather for the first 200, since the selection set contains more than 200 objects). In the Selection view click the gold left-arrow and right-arrow buttons in the header of to cycle visibility between different sets of 200 selected instances at a time.

> **Note**
>
> There will be many more than 200 nets that are re-routed. However, the amount of 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.

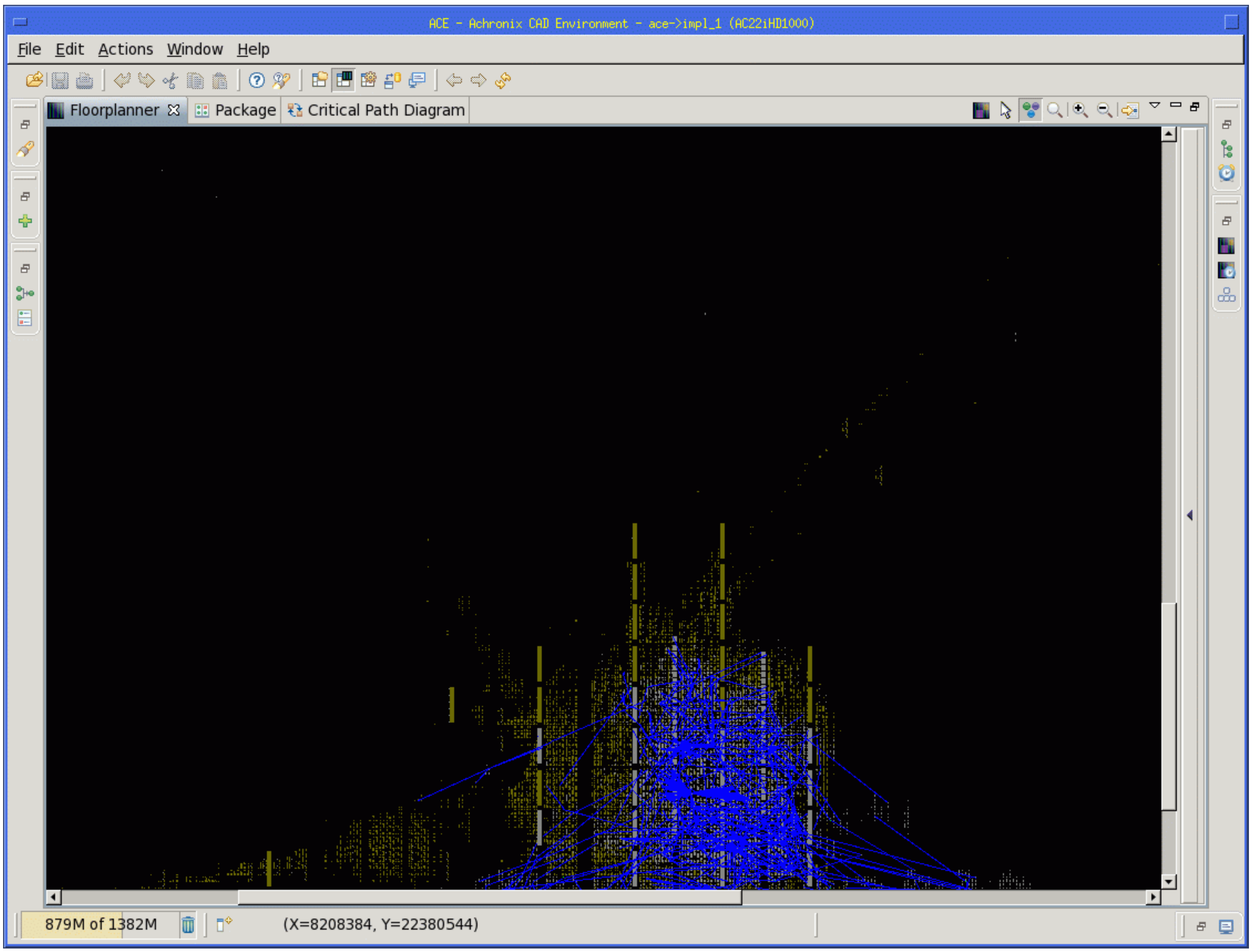


Figure 77: ***ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V1)***

Partitions View (rtl_V1)

In the Partitions View of the Floorplanner Perspective, note that only the top-level partition has 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 those that were not re-placed. This technique can help in understanding the effectiveness of incremental compilation on the design.

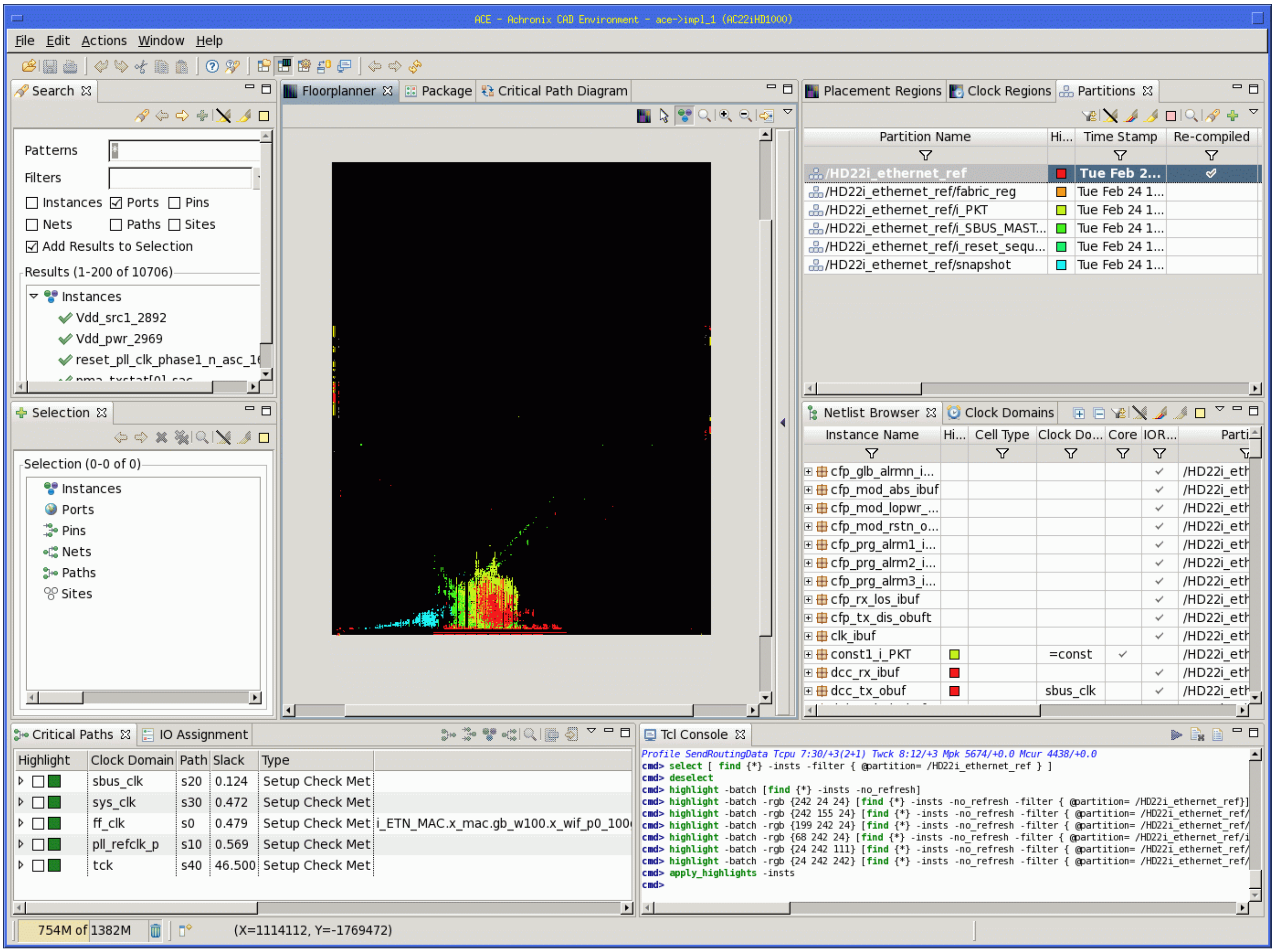


Figure 78: *Highlighting Partitions in ACE After the Second Incremental Compile Iteration*

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 172: ***Additional Tutorial Examples***

Directory	Changes
`rtl_V1`	Removes one signal from SnapShot debugger in `HD22i_ethernet_ref.v`.
`rtl_V2`	Replaces the SnapShot signal removed in `rtl_V1` and adds additional 32-bit flip-flop in `packet_chk.v`.
`rtl_V3`	Adds a counter in `register_control_interface.v`.
`rtl_V4`	Modifies the top-level module by removing inversion from several wires.

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 446), 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 446), 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 will perform an incremental compile and the results will 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.

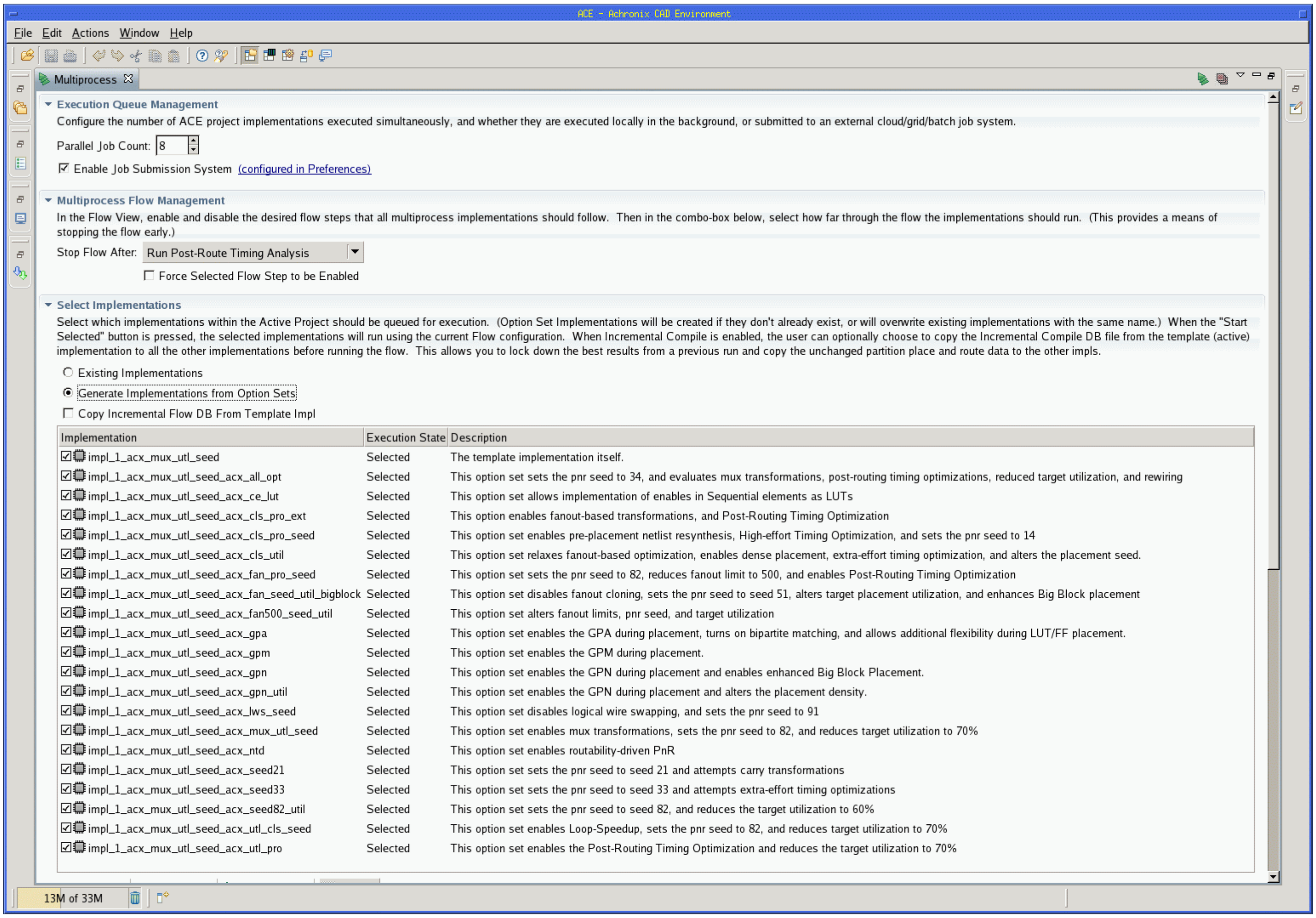
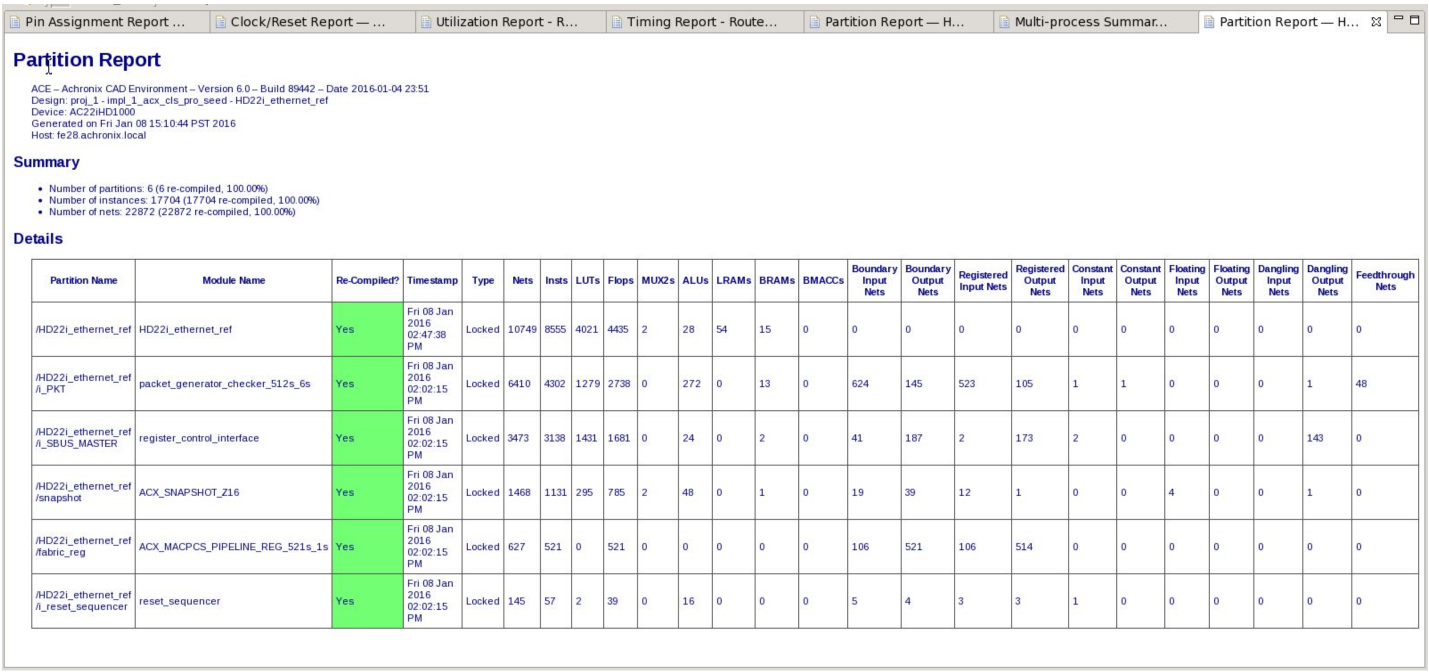


Figure 79: *Multiprocess View Menus*

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.

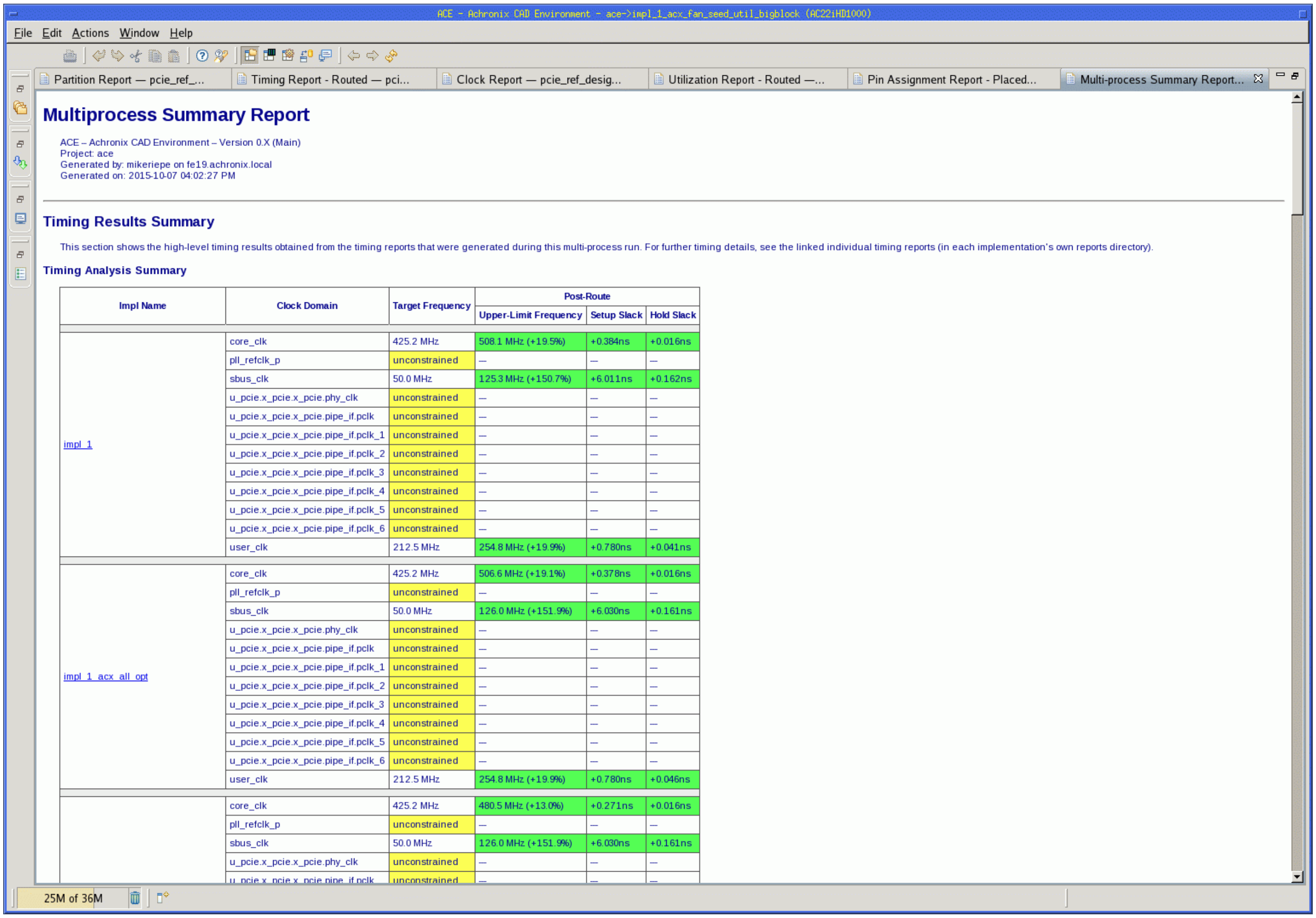


Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX2s	ALUs	LRAMs	BRAMs	BMACCs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthrough Nets
/HD22i_ethernet_ref	HD22i_ethernet_ref	Yes	Fri 08 Jan 2016 02:47:38 PM	Locked	10749	8555	4021	4435	2	28	54	15	0	0	0	0	0	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_PKT	packet_generator_checker_512s_6s	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	6410	4302	1279	2738	0	272	0	13	0	624	145	523	105	1	1	0	0	0	1	48
/HD22i_ethernet_ref /i_SBUS_MASTER	register_control_interface	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	3473	3138	1431	1681	0	24	0	2	0	41	187	2	173	2	0	0	0	0	143	0
/HD22i_ethernet_ref /snapshot	ACX_SNAPSHOT_Z16	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	1468	1131	295	785	2	48	0	1	0	19	39	12	1	0	0	4	0	0	1	0
/HD22i_ethernet_ref /fabric_reg	ACX_MACPCS_PIPELINE_REG_521s_1s	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	627	521	0	521	0	0	0	0	0	106	521	106	514	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_reset_sequencer	reset_sequencer	Yes	Fri 08 Jan 2016 02:02:15 PM	Locked	145	57	2	39	0	16	0	0	0	5	4	3	3	1	0	0	0	0	0	0

Figure 80: ***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	core_clk	425.2 MHz	508.1 MHz (+19.5%)	+0.384ns	+0.016ns
	pll_refclk_p	unconstrained	—	—	—
	sbus_clk	50.0 MHz	125.3 MHz (+150.7%)	+6.011ns	+0.162ns
	u_pcie.x_pcie.x_pcie.phy_clk	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_1	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_2	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_3	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_4	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_5	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_6	unconstrained	—	—	—
	user_clk	212.5 MHz	254.8 MHz (+19.9%)	+0.780ns	+0.041ns
impl_1_acx_all_opt	core_clk	425.2 MHz	506.6 MHz (+19.1%)	+0.378ns	+0.016ns
	pll_refclk_p	unconstrained	—	—	—
	sbus_clk	50.0 MHz	126.0 MHz (+151.9%)	+6.030ns	+0.161ns
	u_pcie.x_pcie.x_pcie.phy_clk	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_1	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_2	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_3	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_4	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_5	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk_6	unconstrained	—	—	—
	user_clk	212.5 MHz	254.8 MHz (+19.9%)	+0.780ns	+0.046ns
	core_clk	425.2 MHz	480.5 MHz (+13.0%)	+0.271ns	+0.016ns
	pll_refclk_p	unconstrained	—	—	—
	sbus_clk	50.0 MHz	126.0 MHz (+151.9%)	+6.030ns	+0.161ns
	u_pcie.x_pcie.x_pcie.phy_clk	unconstrained	—	—	—
	u_pcie.x_pcie.x_pcie.pipe_if.pclk	unconstrained	—	—	—

Figure 81: ***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_fan_seed_util_bigblock (actual results may differ).

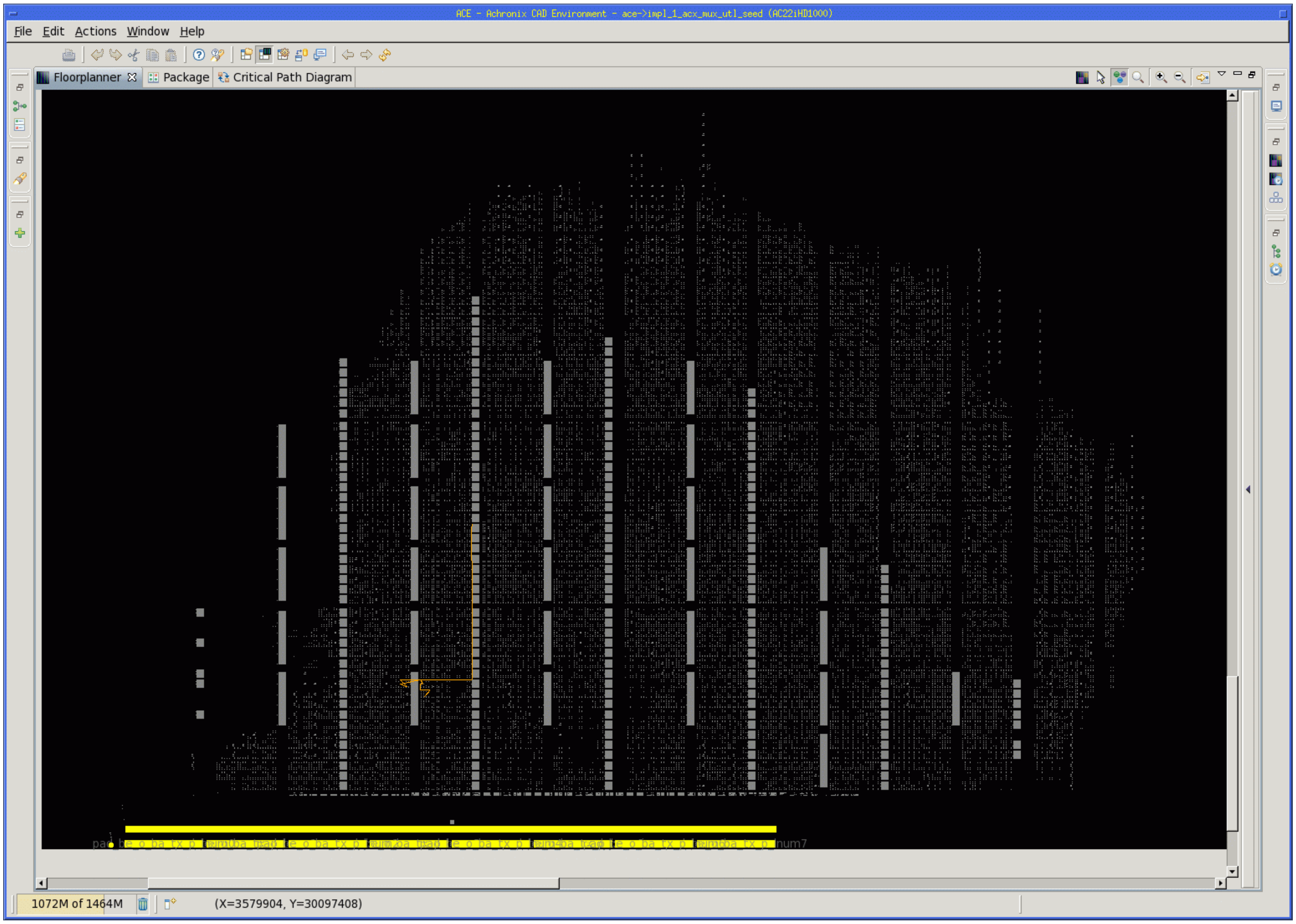


Figure 82: *Critical Path (Orange) in the Best Multiprocess Implementation*

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.

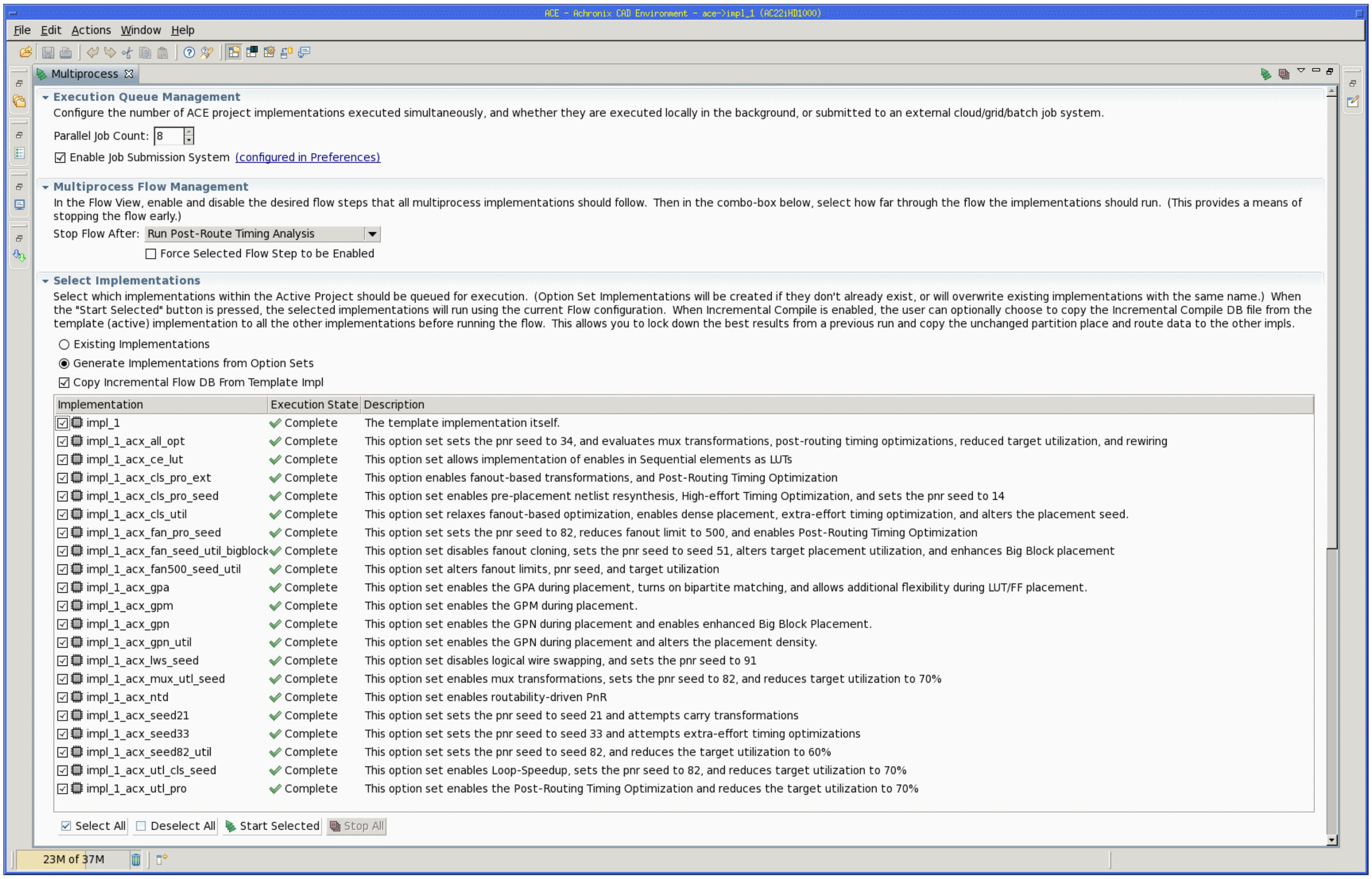


Figure 83: ***Multiprocess View with the Copy Incremental Flow DB from Template Impl Option Checked***

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.

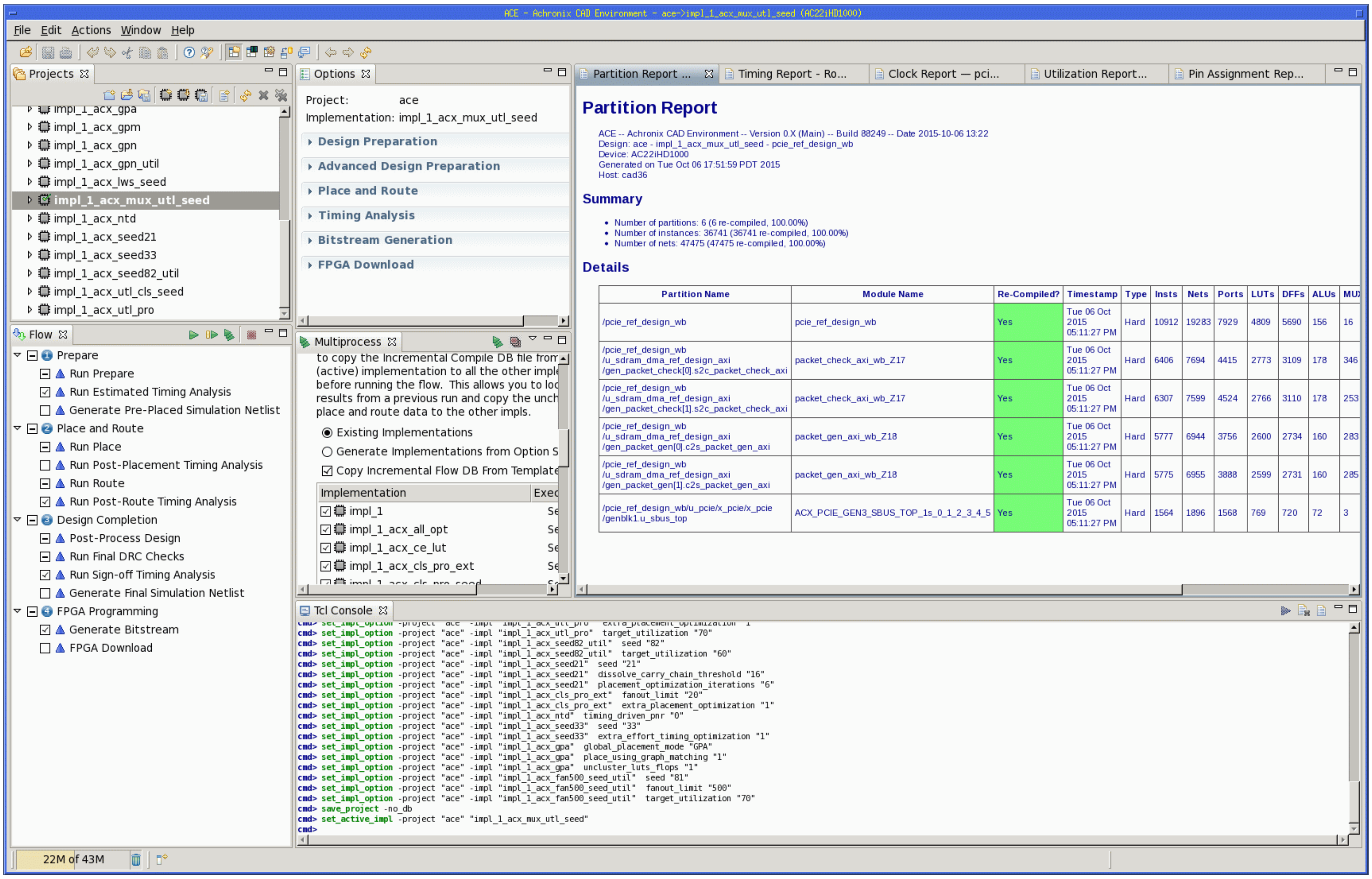


Figure 84: ***Selection of the Template (Active) Implementation***

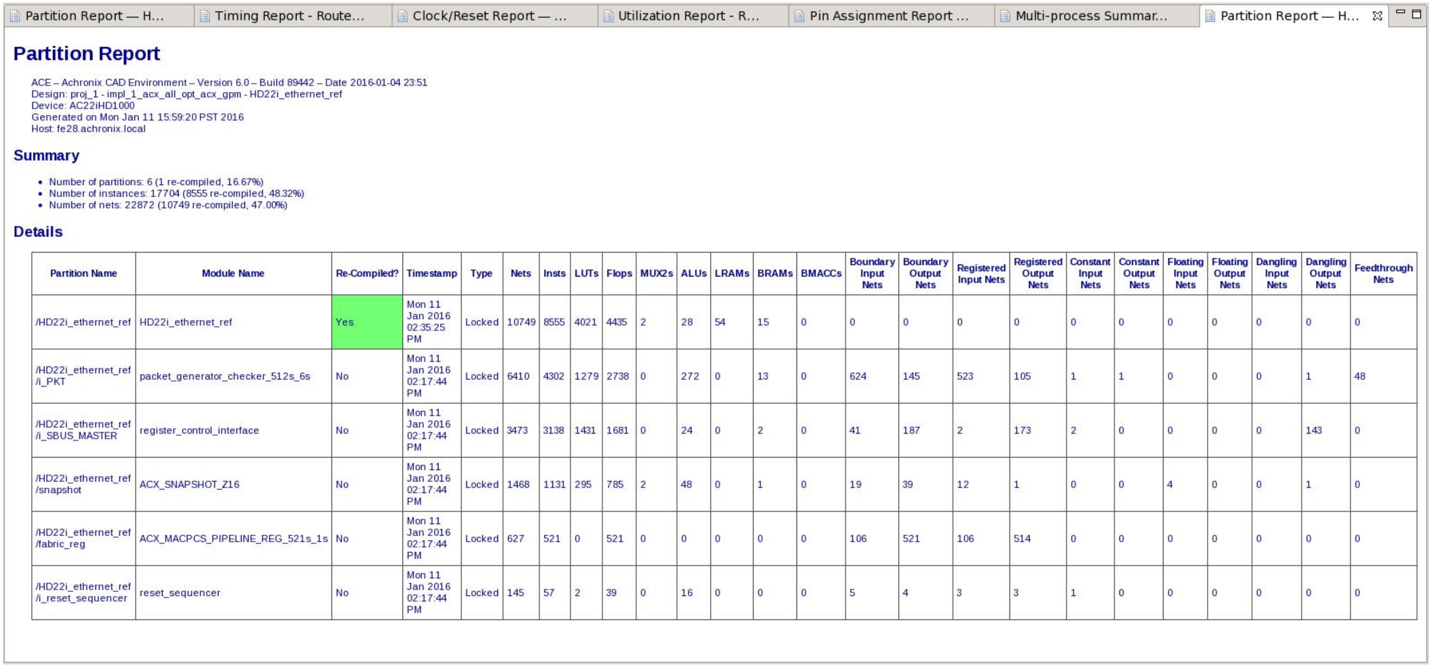
Step 4: Change the RTL and Recompile the Design in Synplify Pro

Repeat Steps 8-10 of the Single-Process Incremental Compile Tutorial (see page 446) 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_all_opt_acx_gpn` implementation, only the single changed partition has been recompiled in this iteration.



Partition Report

ACE – Achronix CAD Environment – Version 6.0 – Build 89442 – Date 2016-01-04 23:51
Design: proj_1 - impl_1_acx_all_opt_acx_gpm - HD22i_ethernet_ref
Device: AC22iHD1000
Generated on Mon Jan 11 15:59:20 PST 2016
Host: fe28.achronix.local

Summary

- Number of partitions: 6 (1 re-compiled, 16.67%)
- Number of instances: 17704 (8555 re-compiled, 48.32%)
- Number of nets: 22872 (10749 re-compiled, 47.00%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX2s	ALUs	LRAMs	BRAMs	BMACCs	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
/HD22i_ethernet_ref	HD22i_ethernet_ref	Yes	Mon 11 Jan 2016 02:35:25 PM	Locked	10749	8555	4021	4435	2	28	54	15	0	0	0	0	0	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_PKT	packet_generator_checker_512s_6s	No	Mon 11 Jan 2016 02:17:44 PM	Locked	6410	4302	1279	2738	0	272	0	13	0	624	145	523	105	1	1	0	0	0	1	48
/HD22i_ethernet_ref /i_SBUS_MASTER	register_control_interface	No	Mon 11 Jan 2016 02:17:44 PM	Locked	3473	3138	1431	1681	0	24	0	2	0	41	187	2	173	2	0	0	0	0	143	0
/HD22i_ethernet_ref /snapshot	ACX_SNAPSHOT_Z16	No	Mon 11 Jan 2016 02:17:44 PM	Locked	1468	1131	295	785	2	48	0	1	0	19	39	12	1	0	0	4	0	0	1	0
/HD22i_ethernet_ref /fabric_reg	ACX_MACPCS_PIPELINE_REG_521s_1s	No	Mon 11 Jan 2016 02:17:44 PM	Locked	627	521	0	521	0	0	0	0	0	106	521	106	514	0	0	0	0	0	0	0
/HD22i_ethernet_ref /i_reset_sequencer	reset_sequencer	No	Mon 11 Jan 2016 02:17:44 PM	Locked	145	57	2	39	0	16	0	0	0	5	4	3	3	1	0	0	0	0	0	0

Figure 85: ***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.

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.

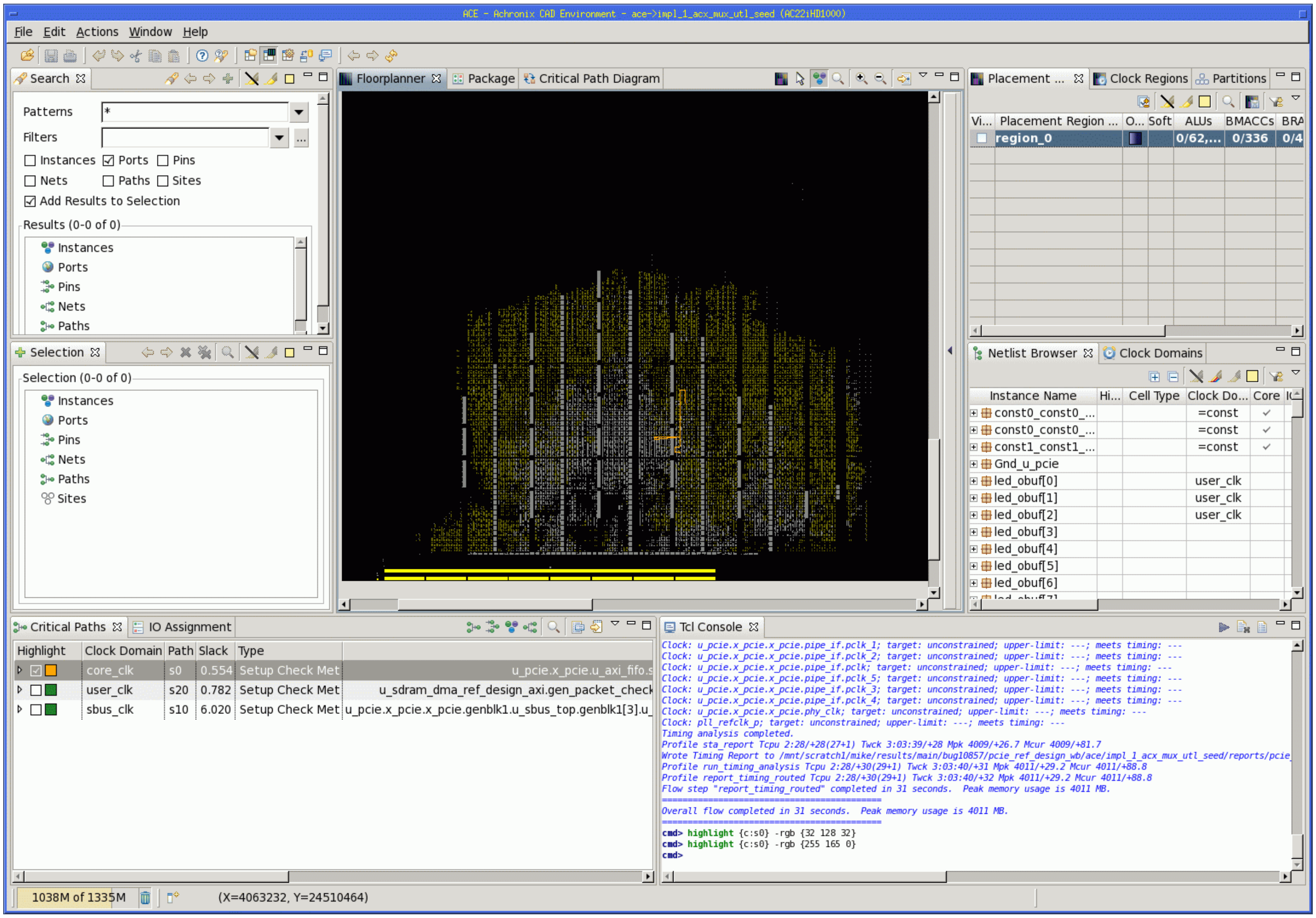


Figure 86: *Critical Path (Orange) from the Best Multiprocess Implementation of the Second Incremental Run*

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. ACE performs this operation in the reconditioner during run_prepare. 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's 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's clock input to the IPIN/OPIN's clock input, and optionally connected the DFF's reset and enable inputs to the IPIN /OPIN's 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 > 1
- 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 below), ACE looks for an attribute named ace_useioff with the following semantics:

- If the ace_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 ace_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 ace_useioff attribute can be placed in several different locations in your RTL code, as described below and shown in following figure.

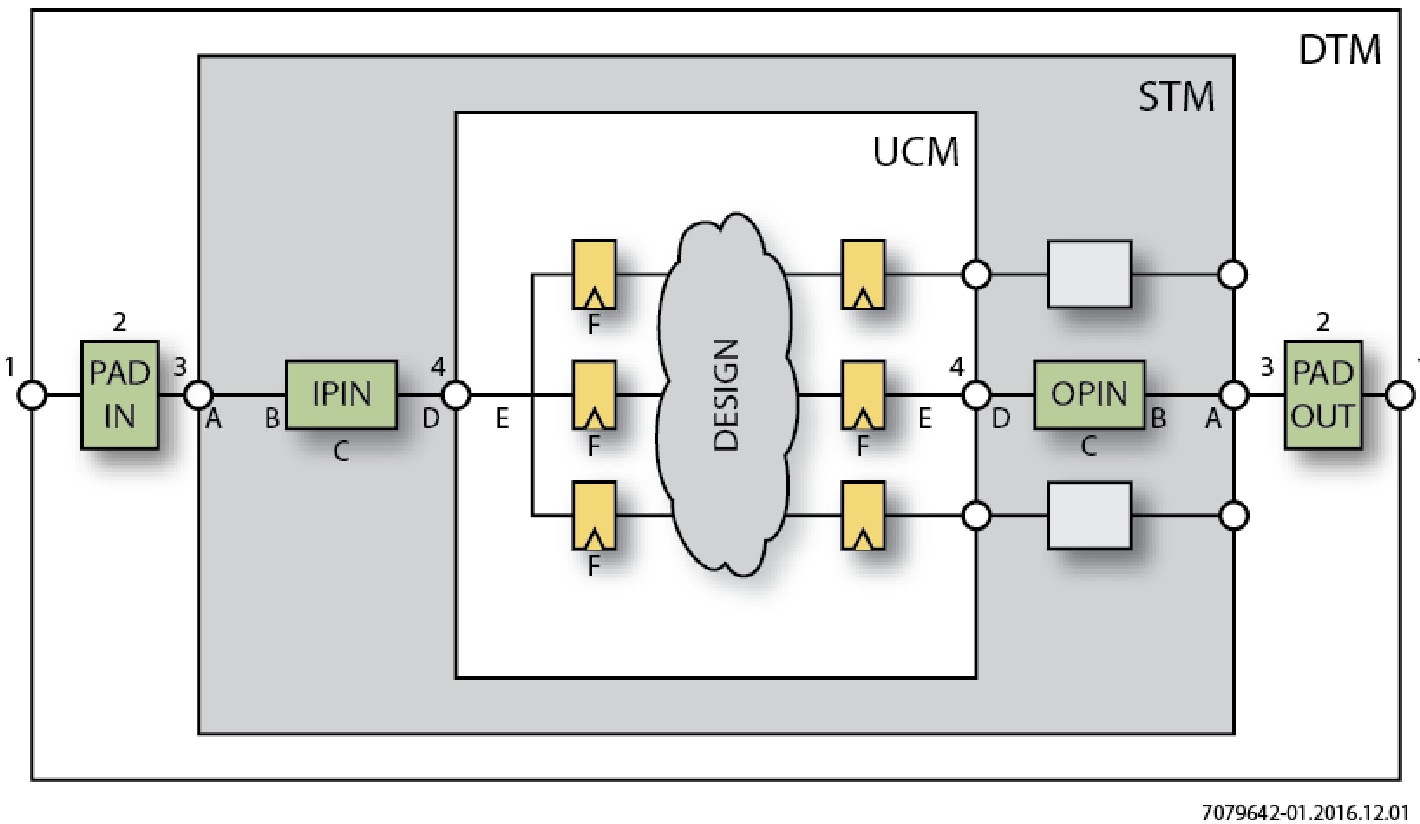


Figure 87: *Valid Locations to Place the ace_useioff Attribute in an eFPGA Design Hierarchy*

- 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 an ace_useioff attribute with the same value.

Specifically, the ace_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. ACE does not trace through the DTM black-box network searching for the `ace_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. 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 `ace_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 modified by the end user. For Speedcore designs in which the STM level of hierarchy is fixed by the ASIC integrator, and only the UCM hierarchy can be modified by the end user, the most convenient locations to place the `ace_useioff` attribute may be the UCM port-to-core nets (E) or the DFF instances (F).

Be careful not to add the `ace_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 `ace_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 the user does not have 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 designer's intent 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

Below are several examples demonstrating how to set the ace_useioff attribute on: top-level ports, I/O pin instances, boundary wires, and DFF instances; and using 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 Speedcore Synthesis User Guide.

Verilog Example of a Port attribute

```
module flop_push_test1 (ina, inb, sel, clk, z0);
    input [3:0] ina /* synthesis ace_useioff=1 */;
    input [3:0] inb /* synthesis ace_useioff=0 */;
    input sel       /* synthesis ace_useioff=1 */;
    input clk;
    output z0       /* synthesis ace_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 ace_useioff : boolean;
attribute ace_useioff of ina : signal is TRUE;
attribute ace_useioff of inb : signal is FALSE;
attribute ace_useioff of sel : signal is TRUE;
attribute ace_useioff of z0 : signal is TRUE;

end entity;
```

Physical Design Constraints (.pdc) File of Port Attributes

```
set_property ace_useioff "1" [find -ports {ina\[*\]}]
set_property ace_useioff "0" [find -ports {inb\[*\]}]
set_property ace_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 484).

It is not possible using RTL attributes to apply different values of the `ace_useioff` attribute to different ports in a port bus (for example, giving in_a[2] an `ace_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 example flop_push_test5 below). The solution requires bit-blasting the bus into separate ports or assigning different values of `ace_useioff` to different port 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 ace_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 ace_useioff "1" [find -insts {ipin_*}]
```

```
set_property ace_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 ace_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 ace_useioff "1" [find -nets {ipin_dout_*}]
set_property ace_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 ace_useioff = 0 */;
    ACX_DFF dff2 (.d(ipin_dout_37), .clk(clk), q(dff2_q))  /* synthesis ace_useioff = 0 */;
endmodule
```

```
Physical Design Constraints (.pdc) File of DFF Instance Attributes

set_property ace_useioff "1" [find -insts {dff*}]
set_property ace_useioff "0" {i:dff1 i:dff2}
```

Below 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 ace_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 `ace_useioff` attribute to different ports in a port bus. The above example flop_push_test5 appears to be a clever way to apply the `ace_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 ace_useioff attribute is not forward annotated into the ACE input netlist. One can use PDC, however, to apply the `ace_useioff` attribute to the IPIN instance as in the example below.

```
Physical Design Constraints (.pdc) For Example flop_push_test5 That DOES Work

set_property ace_useioff "1" [find -nets {in\[*\]}]
set_property ace_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 below. See also Options View (see page 215) in the ACE User Guide.

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 `ace_useioff` attribute set to "1"
- “15" – (automatic mode) push flops into all pads *except* those that have the `ace_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. It 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_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 will be 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's input by reducing the wiring delay. But it increases the on-chip timing path at the flop's 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 is 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. A user in not permitted to export a bitstream for a design with virtualized I/Os. If the design has 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/Os, 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. The user can also manually select pin buses and individual pins for virtualization through top-level port attributes in their RTL or PDC constraints (see Port Attributes (see page 492) 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's output net. Similarly, each OPIN is replaced with a “stub” LUT, driven by the OPIN's input net, with a floating output pin. These stub LUTs are given `must_keep` attributes so that ACE will 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 will pull 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 will only see 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) will contribute 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 specified by the user 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 491) and Port Attributes (see page 492).

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 enum values are: **stubout** (the default), **serialize_dff**, and **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 will fit in 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 will be 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, users can manually control which port buses are virtualized through the use of the RTL port attribute `ace_virtualize`. When the virtualization style is set to `serialize_dff`, one can also specify either a top-level port name or net name to be connected to the clock input of the new serialization flop instances. 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

Each of the available virtualization styles are illustrated below 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.

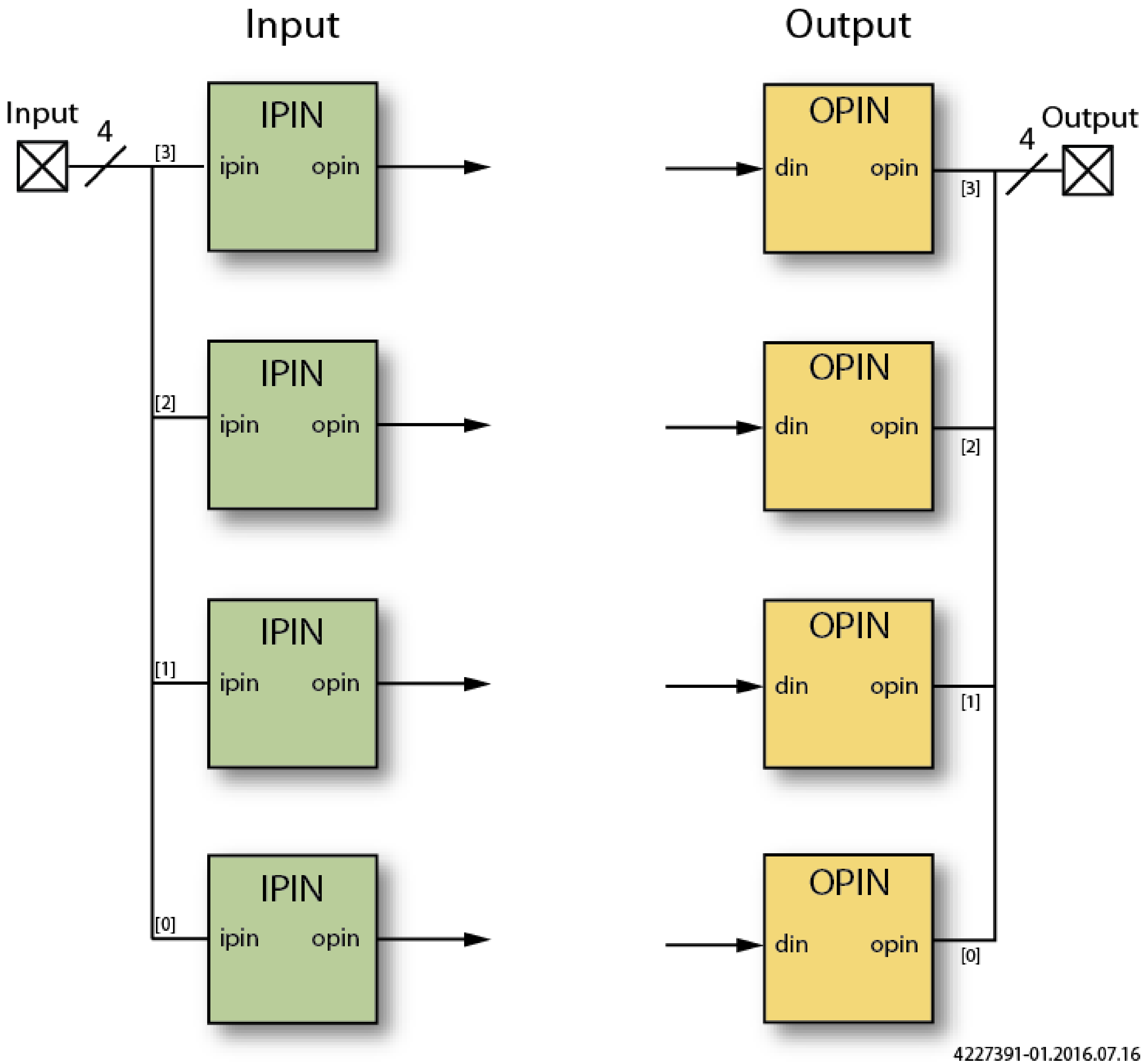


Figure 88: ***Input and Output Pads***

Output Netlist Styles

stubout

The following two schematics illustrate the output of IO 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.

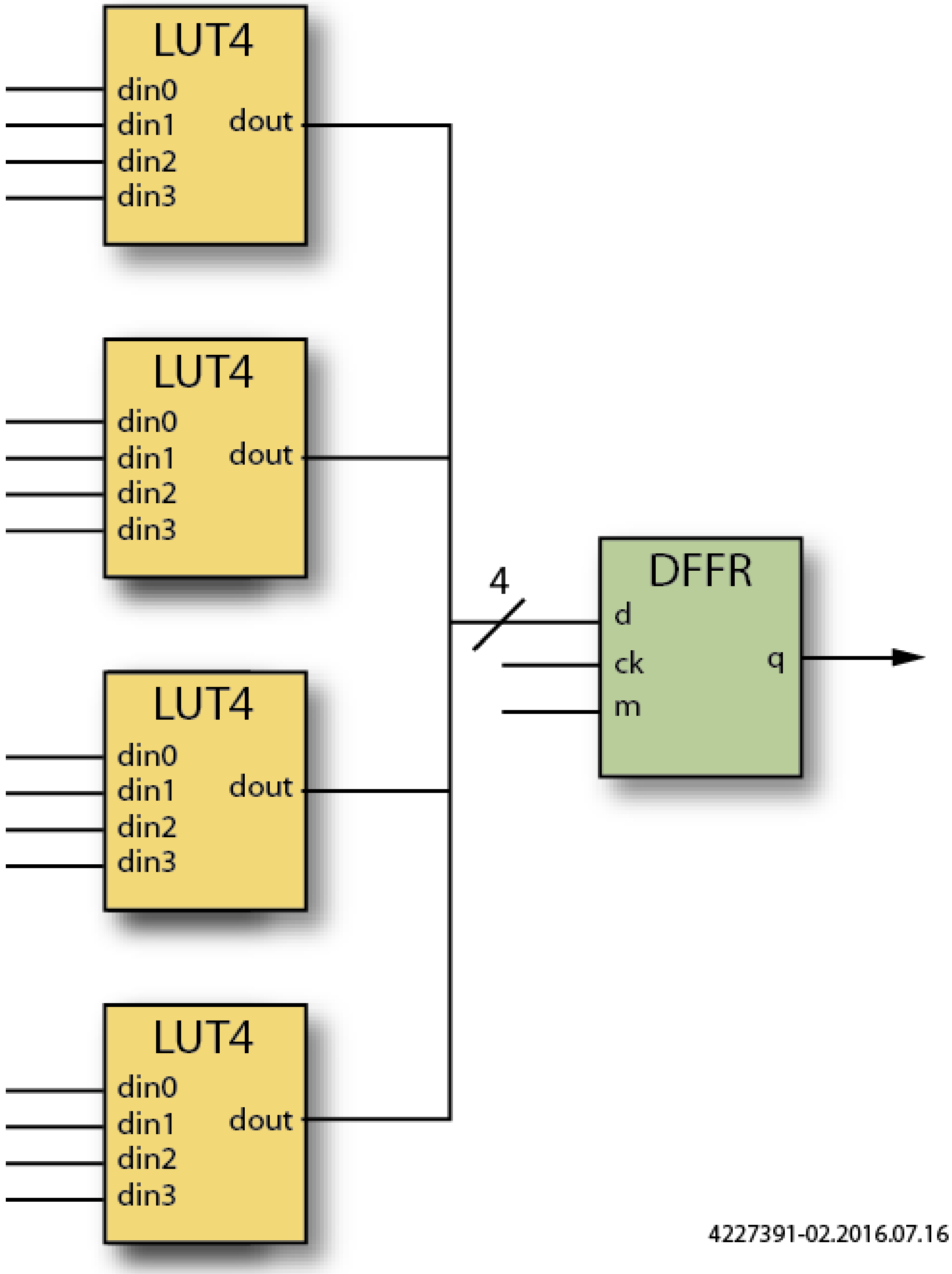


Figure 89: ***Stubout Style Input Pad***

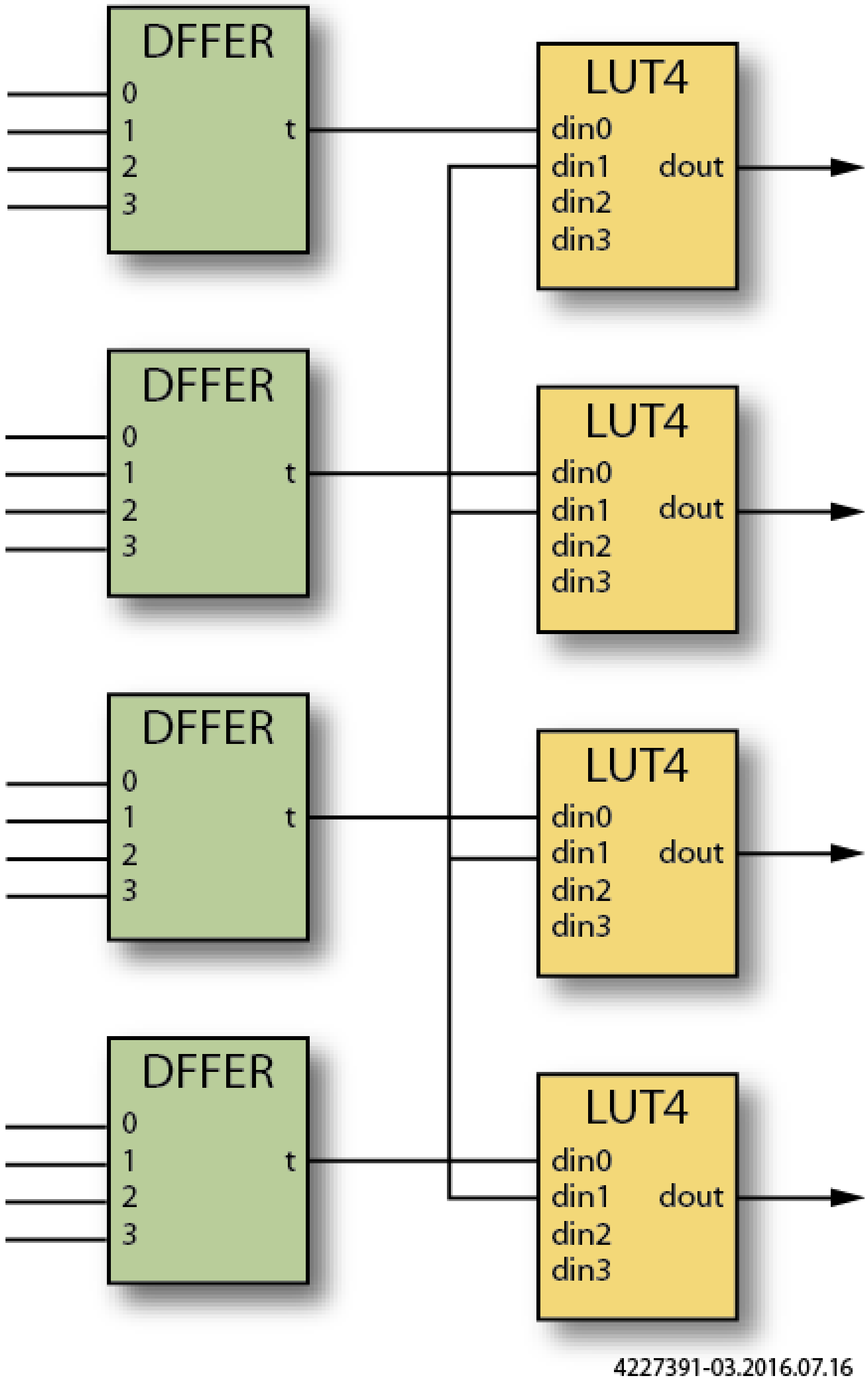


Figure 90: ***Stubout Style Output Pad***

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, note that the 4-bit output DFF shift chain is driven by 4 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.

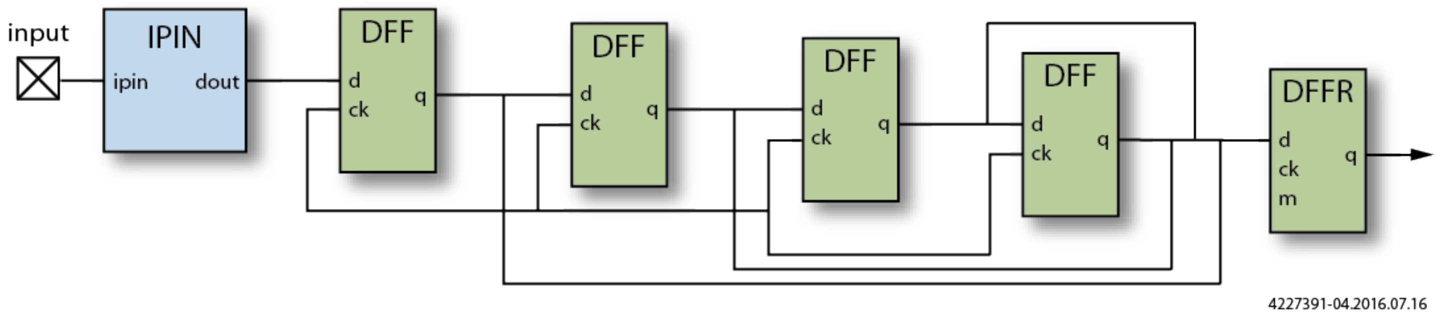


Figure 91: ***serialize_dff Style Input Pad***

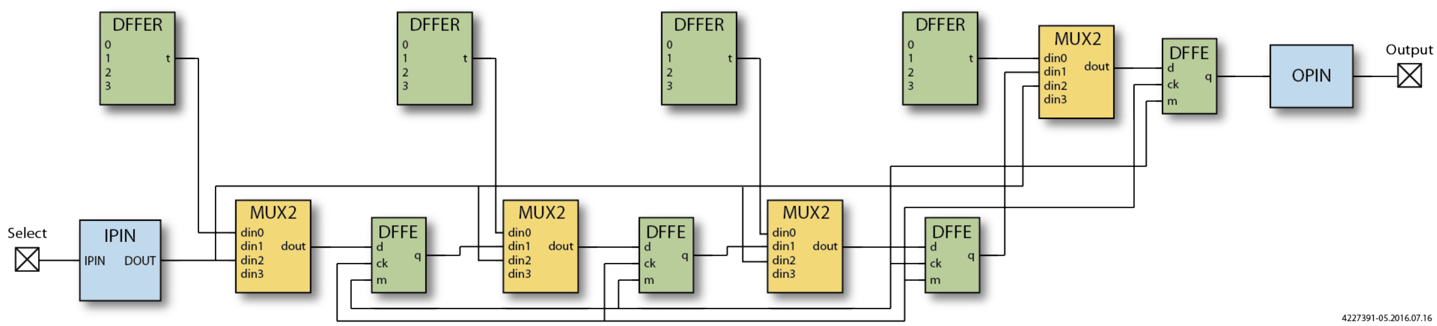


Figure 92: ***serialize_dff Style Output Pad***

serialize_lut

The following two schematics illustrate the output of IO 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, note that the 4-bit output LUT shift chain is driven by 4 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.

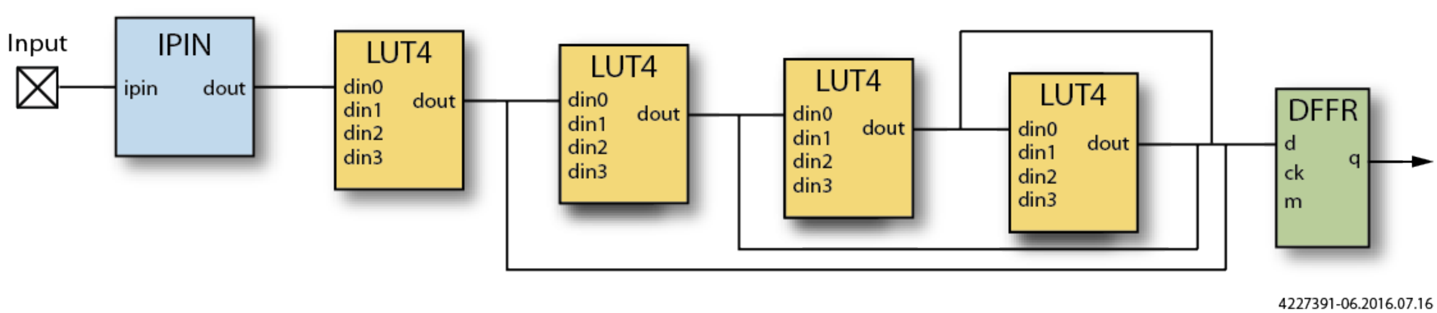


Figure 93: ***serialize_lut Style Input Pad***

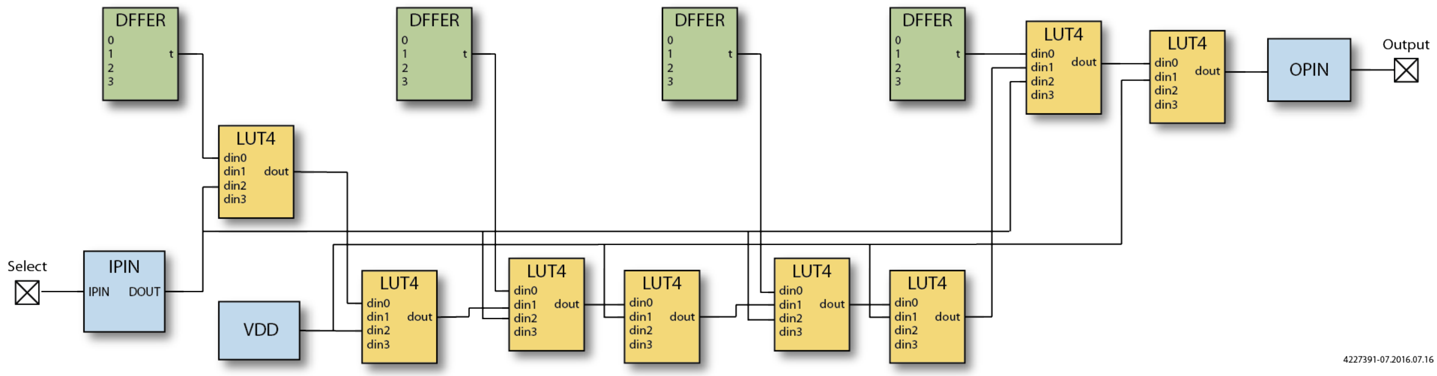


Figure 94: *serialize_lut Style Output Pad*

Viewing the Package Layout

> **The Package View is only applicable for Speedster FPGA devices**
>
> Users should ignore the Package View (see page 226) when developing for other Achronix product types.

Opening and Closing the Package View's Fly-Out Palette

To open and close the Package View (see page 226)'s fly-out palette of view options:

1. Click on the **Fly-out** button (◀) on the far right side of the Package View (see page 226) 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. Once the view options are configured, click on the Fly-in button (▶) on the left side of the fly-out palette to close the fly-out palette.

Zooming the Package In and Out

There are several ways to zoom in and out in the Package view (see page 226).

Note:	*Zoom levels are always in powers of 2, i.e. zoom in is at 200% and zoom out is at 50%. Therefore, it may 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 over the point for zoom in or out from in the Package view.
2. Slide the mouse wheel forward to zoom in or slide the mouse wheel backward to zoom out.

To zoom in and out with key-strokes:

1. Hover the mouse over the center of the area for zoom in or out from in the Package 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 and drag the mouse to the lower right until the zoom rectangle encloses the area desired. To zoom out, click the point on the Package view to zoom out from 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 over the point to zoom in or out from in the Package view. Single-click the left mouse button to zoom in or single-click 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 to zoom in to or out from in the Package view.
2. Press the **Zoom In** button ( ) to zoom in or the **Zoom Out** button ( ) to zoom out.

Package 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.

To pan with key-strokes:

1. Use the **ARROW** keys on the keyboard to pan left, right, up and down.

To pan with the **Placement Tool**:

1. Select the **Placement Tool** ( ) from the view toolbar.
2. Hover over any point in the Package view which shows the **Pan** cursor ( ). Click and drag the view with the mouse to pan around.

Selecting Package Objects

To select objects with key-strokes:

1. In the **Selection** ( ) section of the 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 position to set the current selection, or press and hold the '**SHIFT+S**' keys to start a selection rectangle at the current mouse position to add to the current selection.
3. Drag the mouse while holding down the key or keys on the keyboard to create a selection rectangle includes the objects desired. Then, release the keys to apply the selection.

To select objects with the Selection Tool:

1. Select the **Selection Tool** ( ) from the view toolbar. From the **Selection** section of the fly-out palette, check the object types you wish to deselect. Also, ensure the Action control is set to **Select**.

2. 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.
   Note: *Not using* **CTRL** *will clear the previous selection.*
3. Release the mouse button to apply the selection.

Deselecting Package Objects

To deselect objects with key strokes:

1. Select the **Selection Tool** ( ) from the view toolbar. From the **Selection** section fly-out palette, check the object types to deselect.
2. Press and hold the '**d**' key on the keyboard to start a selection rectangle at the current mouse position to deselect the objects.
3. Drag the mouse while holding down the key to create a selection rectangle including the objects to deselect. Then, 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. From the **Selection** section of the fly-out palette, check the object types to deselect. Also, ensure the Action control is set to **Deselect**.
2. Click and drag with the left mouse button in the view to create a selection rectangle. Then, release the mouse button to remove the objects from the current selection set.

Toggling Package Mouse Tools

To toggle the mouse tools for the Package View (see page 226):

1. Press the **ALT** key on the keyboard to switch between tools, or simply click on the desired mouse tool on the view toolbar.

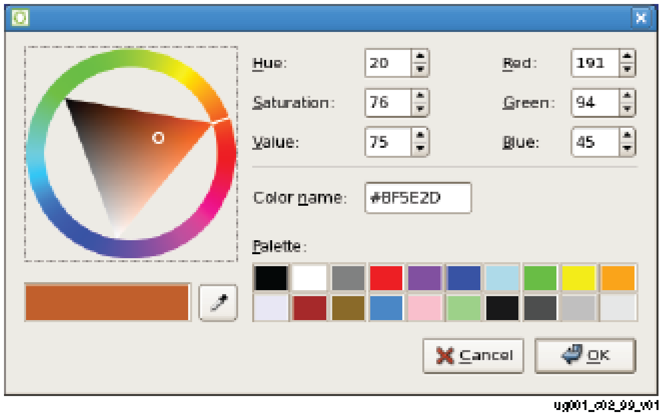
Filtering the Package View

It may be useful at times to filter the view graphics to see only ball locations of interest. Filtering of the view is done with **Layer** ( ) options. When a layer is turned off, selected objects in the current ACE selection set remain visible. For example, un-check **Instances** to see just the selected instances. Alternately, the visibility of individual ball groups can be turned on and off by checking/un-checking the ball-group name. The visibility of all ball groups can be toggled via the **Toggle Groups** button.

To filter the layers in the Package 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 either instances or ball groups to hide.The color coding for each ball group can be altered by clicking the color button ( ) (the colored buttons to the right of each ball group in the **Layers** section of the fly-out palette) to open the color change dialog box. The dialog box provides several options for altering the display color for each ball group.

Changing Color Coding



Note that the dialog is the common color chooser dialog from your operating system - the dialog from Linux is shown above.

Getting Package Object Tooltips

For instant feedback on instance or site names in the Package View (see page 226), a tooltip (hover text) can be enabled. In addition, the contents of the tooltip can be printed to the Tcl Console View (see page 258) for easy copy and paste.

To get object tooltip text:

1. In the **Tool Tip Text** ( ) section of the fly-out palette, select the checkboxes of the object types for which tool tip text is desired.
2. In the Package view, hover over objects to display tool tip text.

> Optionally, press the '**p**' key on the keyboard while the 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 Package Object Labels

A variety of object labels are available when displaying objects in the Package View (see page 226) (see “Fly-Out Palette”).

To display object labels in the Package view:

1. In the **Labels** ( ) section of the fly-out palette, check the object labels to display.
2. Pan and zoom to objects of interest to view the object labels.

Note: *Some labels do not show up unless the view is zoomed in far enough to display the extent of the text.*

Managing I/Os

> **The IO Assignment View is only applicable for Speedster FPGA devices**
>
> Users should ignore the IO Assignment View (see page 191) when developing for other Achronix product types.

I/O electrical properties are often iteratively tweaked at the final stages of design. Frequently the user does not want to alter their source RTL to make these changes, because doing so would necessitate re-running the entire Flow.

The IO Assignment View (see page 191) was created to ease these last-minute tweaks. This view allows I/O electrical changes to be made without impacting the PnR. The user can iteratively tweak electrical settings, regenerate the Bitstream, and test, repeating until the design performs as desired. The user can then save off the set of changed property values in an `.sdc` file (see Save Changed Properties Dialog (see page 280)). 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 IO Assignment View (see page 191), 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 will help ensure that the user does not create 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 Microsoft Windows, or `Shift+F1` in Linux, and the contextual help will appear in a view on the right.

Below is an example of what appears when contextual help is opened while the Projects View (see page 238) has focus:

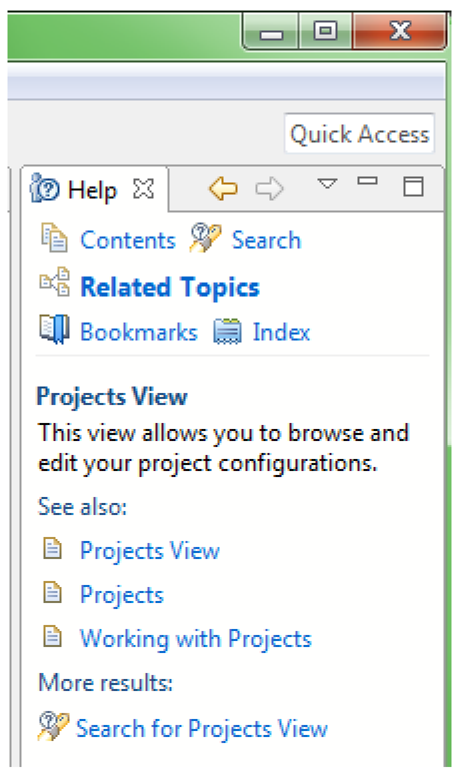


Figure 95: ***Context-Sensitive Help Example***

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 a window 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 currently a known bug (***Linux-only***) in the application frameworks underlying ACE that may cause view /editor tab movements to detach instead of docking when the Help Window is open. See the Troubleshooting (see page 602) section for more details, including several workarounds.

Navigating the Help Window

Table of Contents

1. In the left frame, select the **Contents** tab.
2. Find the topic to be read in the table of contents by clicking to expand the subtopics, then 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 504).

Synchronizing

Clicking the **Show in Table of Contents** button above the right frame selects that page's topic 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 on any blank part of the frame's toolbar. 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 23) 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 users find desired information. To search help:

1. From the main menu, select **Help → Search.**
2. Type in the word or phrase to search for.
3. Click **GO** or press **Enter**. The list of results are displayed below in the left frame (within the **Search Results** tab).
4. To view the content of a topic in the list of results, click it.

Alternatively, 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

ACE users at sites licensed for both Speedcore and Speedster devices can narrow the **Scope** of the search by restricting search scopes to only the user guide(s) they prefer to use.

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 pressed, causes the results to be grouped by book (this action will only have 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 pressed, 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 the following search expression rules for searching local help content:

- The following stop words are common English words which will be 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. For example:
  `Workbench`
  returns topics that contain 'workbench', 'Workbench', 'WorkBench', and 'WORKBENCH'.
- Unless otherwise stated, there is an implied AND between all search terms. In other words, topics that contain all the search terms will be returned. For example:
  `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. For example:
`project OR implementation`
returns topics that contain the word 'project' or the word 'implementation' (or both).
- Use NOT before terms you want to exclude from search results. For example:
`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. For example:
`par?`
returns topics that contain 'part' or 'park', but not 'participate'. On the other hand:
`par*`
returns topics that contain 'part', 'park', 'participate', 'pardon', and so on.

> **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. For example:
`"creating projects"`
returns topics that contain the entire phrase 'creating projects'. Topics where the words 'creating' and 'projects' are not consecutive are not returned.
- Punctuation acts as term delimiters. For example:
`plugin.xml`
returns hits on topics that contain 'plugin' and 'xml', which is likely broader than is typically desired. To find just those topics containing 'plugin.xml', use double quotes in the search, as in:
`"plugin.xml"`

> **Note**
>
> The search engine automatically does "fuzzy" searches and word stemming. Entering
> `create`
> returns results including hits on topics that contain 'creates', 'creating', 'creator', and so on. To prevent the search engine from stemming a term, enclose the term in double quotation marks.

Importing and Exporting Preferences

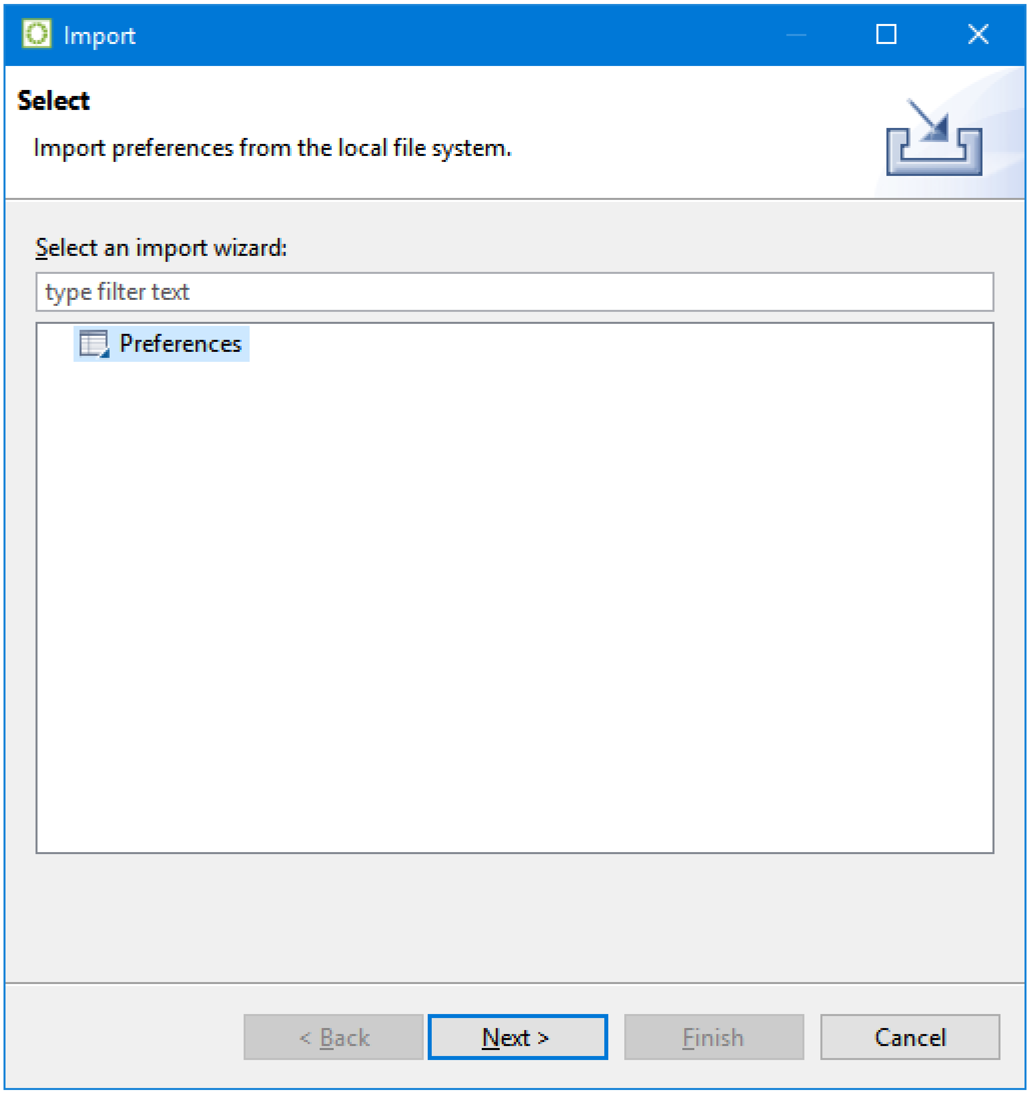
Preference files can be both imported to and exported from ACE, allowing individual or group preferences to be shared or migrated from an existing version of ACE to a newer version when upgrading.

Import Preferences

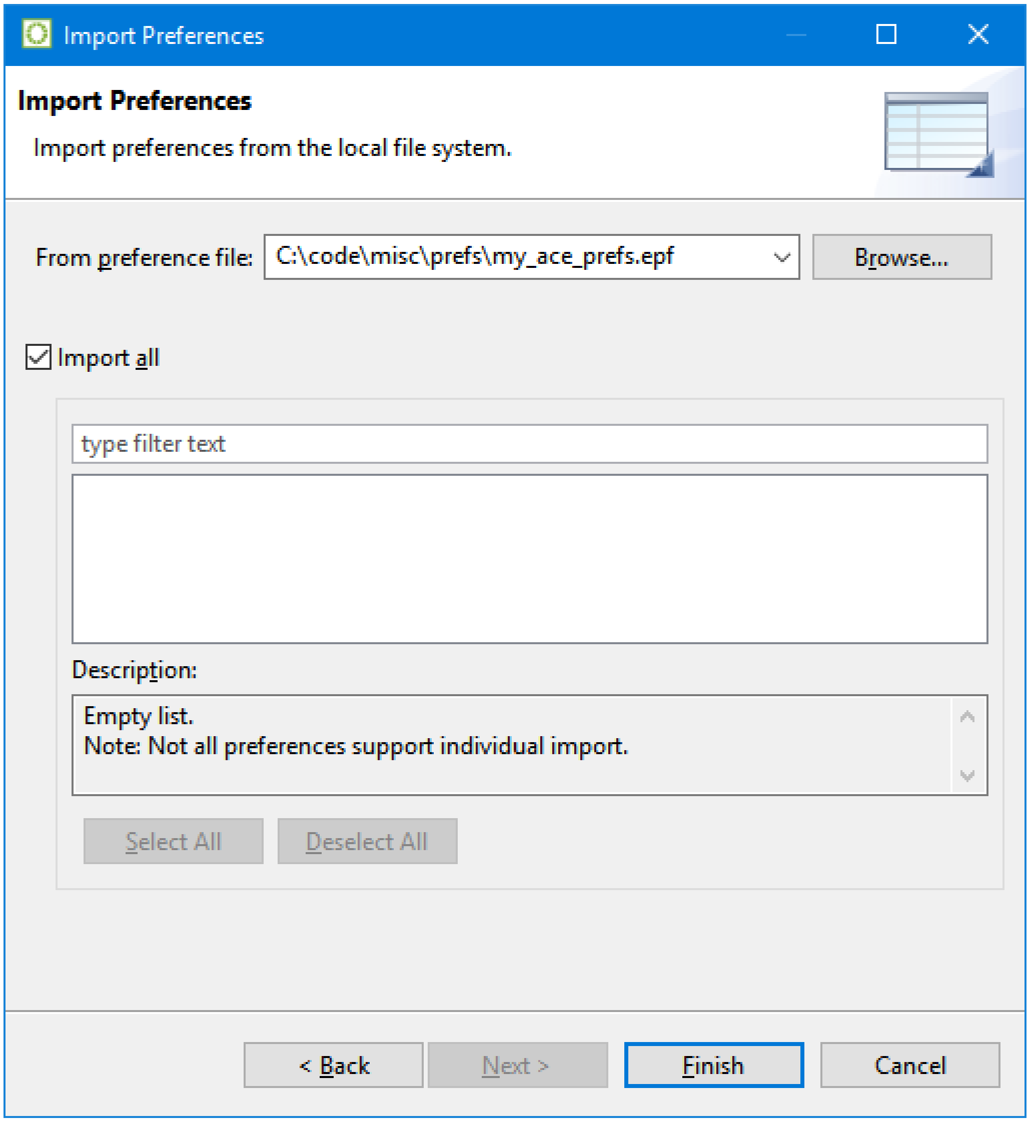
The Import wizard can be used to import preferences from the file system into ACE. To import a preference file:

1. Select **File | Import...**

2. In the Import wizard select **Preferences** and click **Next**.



3. Click **Browse...** and locate the Preferences file on the file system.

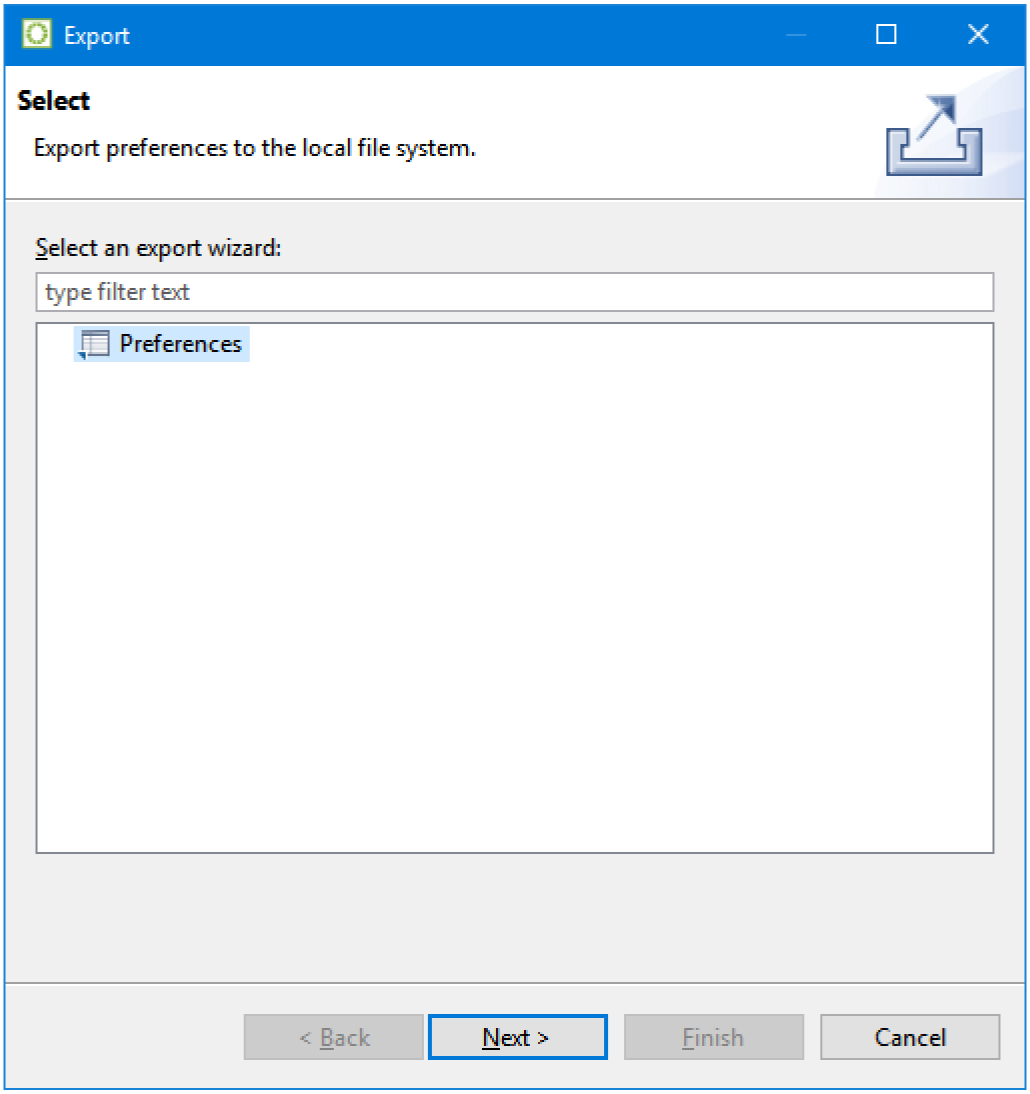


4. Click **Import all** to accept all of the preferences defined in the file.
5. Click **Finish**.

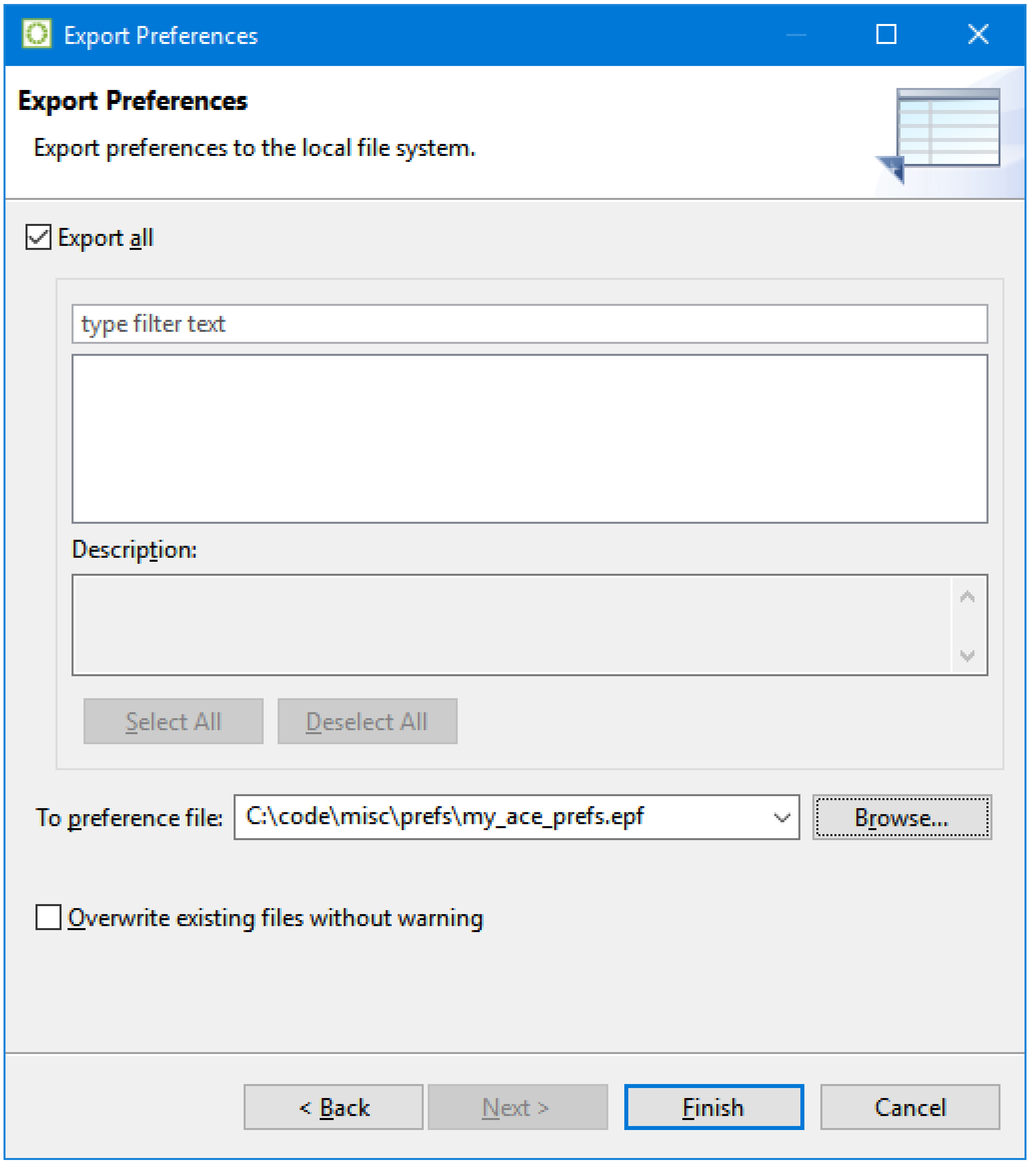
Export Preferences

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**.



3. Click **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**

Chapter - 4: Tcl Command Reference

The Tcl commands supported by ACE are broken into three subsets in this document:

- The SDC Commands (see page 510), 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 529), 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 535), 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 (see page 511)

create_generated_clock (see page 513)

get_clocks (see page 514)

set_clock_uncertainty (see page 520)

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 (see page 524)

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 (see page 528)

create_clock

```
create_clock [<clock>] [-period <string>] [-name <string>] [-waveform <list>]
```

Define a clock

Argument	Optional	Description
[<clock>]	✓	nets, ports or pins (as 'inst/pin')
[-period <string>]	✓	clock period in ns (required)
[-name <string>]	✓	alternate name
[-waveform <list>]	✓	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 (see page 528) constraint.
2. Traverse from the create_clock statement, through a set_input_delay (see page 524) 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 (see page 513)

get_ports (see page 517)

report_clock_properties (see page 533)

report_clocks (see page 573)

report_checks (see page 531)

set_clock_latency (see page 518)

set_clock_groups (see page 517)

set_clock_uncertainty (see page 520)

set_input_delay (see page 524)

set_output_delay (see page 528)

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>]`	✓	(required) net or pin
`[-divide_by <int>]`	✓	factor
`[-multiply_by <int>]`	✓	factor
`[-name <string>]`	✓	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 (see page 511) definition, of which the port used to define that create_clock (see page 511) 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/-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 (see page 511)

get_ports (see page 517)
report_clock_properties (see page 533)
report_clocks (see page 573)
report_checks (see page 531)
set_clock_groups (see page 517)
set_clock_latency (see page 518)
set_clock_uncertainty (see page 520)

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 of the cell instances with the string "reg" in it, enter;

```
get_cells *reg*
```

The output of the [get_cells] command can be passed to other commands, such as [set_multicycle_path]. Here all timing paths from the cell instances (top.*my_module.module_reg*

```
set_multicycle_path -from [get_cells top.*my_module.module_reg*] -setup -end 2
```

Also See

set_multicycle_path (see page 527)
set_false_path (see page 523)

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]`	✔	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 (see page 511) command is used to define clocks. Additionally, if the create_generated_clock (see page 513) 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 (see page 510) 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 (see page 511)

create_generated_clock (see page 513)

report_clock_properties (see page 533)

report_clocks (see page 573)

set_clock_groups (see page 517)

set_clock_latency (see page 518)

set_clock_uncertainty (see page 520)

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 of the nets in the design that have the string "data" in them as well as the string "[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 (see page 513)

get_clocks (see page 514)

set_false_path (see page 523)

set_multicycle_path (see page 527)

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/*reg*/d] -setup 4
```

Also See

create_generated_clock (see page 513)

set_false_path (see page 523)

set_multicycle_path (see page 527)

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 of the ports with the sting "[0]" in it, 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 (see page 511)

create_generated_clock (see page 513)

set_false_path (see page 523)

set_input_delay (see page 524)

set_multicycle_path (see page 527)

set_output_delay (see page 528)

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>]	✔	Name of clock group
[-group <list>]	✔	set of clocks
[-asynchronous]	✔	clocks are unrelated (default)

Description

The set_clock_groups command is defined in the SDC files after the create_clock (see page 511) and create_generated_clock (see page 513) statements have been defined. This command can be used to quickly define asynchronous relationships between clocks. This methodology replaced the older set_false_path (see page 523) 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 (see page 511)

create_generated_clock (see page 513)

get_clocks (see page 514)

report_checks (see page 531)

set_false_path (see page 523)

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

Argument	Optional	Description
`port_pin_list`		port_pin_list (one or more ports)
`[-clock <string>]`	✔	clock list
`[-rise]`	✔	rise
`[-fall]`	✔	fall
`[-min]`	✔	min
`[-max]`	✔	max
`[-late]`	✔	late
`[-early]`	✔	early
`[-source]`	✔	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 (see page 511) 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 (see page 531) 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 (see page 520) 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 (see page 511)

get_clocks (see page 514)

report_clock_properties (see page 533)

report_clocks (see page 573)

report_checks (see page 531)

set_clock_uncertainty (see page 520)

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>]`	✔	one or more clocks, ports, or pins
`[-from <string>]`	✔	source clock
`[-to <string>]`	✔	destination clock
`[-setup]`	✔	uncertainty applies to setup check
`[-hold]`	✔	uncertainty applies to hold check

Description

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 create_clock (see page 511)'s definitions, as well as set_clock_latency (see page 518) values. Can should be taken to ensure that uncertainty defined in those other constraint commands are not duplicated in the set_clock_uncertainty command. The effects 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 (see page 531) 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 (see page 510)
create_clock (see page 511)
create_generated_clock (see page 513)
report_clock_properties (see page 533)
report_clocks (see page 573)
report_checks (see page 531)
set_clock_latency (see page 518)

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>]`	✔	clock
`[-setup]`	✔	setup
`[-hold]`	✔	hold
`[-from <string>]`	✔	from_list (one or more clocks)
`[-to <string>]`	✔	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 or -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 (see page 528) 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 (see page 528) constraints define.

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 (see page 516)

get_clocks (see page 514)

report_checks (see page 531)

set_output_delay (see page 528)

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>]	✓	input pin name of instance <object>
[-to <string>]	✓	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 (see page 523), 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 it's effect. Often, this command is used to break timing loops from an input pin to an output pins 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 (see page 514)

set_false_path (see page 523)

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>]`	✔	from_list (one or more clocks)
`[-to <list>]`	✔	to_list (one or more clocks)
`[-through <list>]`	✔	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 (see page 527) 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 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 int he 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 "pealing an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next worst paths becomes the 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

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 (see page 516)

get_ports (see page 517)

get_nets (see page 515)

set_multicycle_path (see page 527)

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)

Argument	Optional	Description
`[-clock <string>]`	✔	clock_name
`[-rise]`	✔	rise
`[-fall]`	✔	fall
`[-max]`	✔	max
`[-min]`	✔	min
`[-add_delay]`	✔	add delay
`[-clock_fall]`	✔	delay with reference to falling edge of clock

Description

The set_input_delay, as well as the set_output_delay (see page 528) 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 (see page 518) 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 (see page 531) 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 (see page 511)

get_clocks (see page 514)

get_ports (see page 517)

report_checks (see page 531)

set_clock_latency (see page 518)

set_output_delay (see page 528)

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>]	✔	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>]	✔	from_list (one or more clocks)
[-to <list>]	✔	to_list (one or more clocks)
[-through <list>]	✔	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

Argument	Optional	Description
[-from <list>]	✔	from_list (one or more clocks)
[-to <list>]	✔	to_list (one or more clocks)
[-through <list>]	✔	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		multiplier
[-setup]	✔	setup
[-hold]	✔	hold
[-start]	✔	start
[-end]	✔	end
[-from <list>]	✔	from_list (one or more clocks)
[-to <list>]	✔	to_list (one or more clocks)
[-through <list>]	✔	through_list (one or more clocks)

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 int he 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 "pealing an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next worst paths becomes the 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 a 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
```

In 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 (see page 516)

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>]`	✔	clock_name
`[-rise]`	✔	rise
`[-fall]`	✔	fall

Argument	Optional	Description
[-max]	✔	max
[-min]	✔	min
[-add_delay]	✔	add delay
[-clock_fall]	✔	delay with reference to falling edge of clock

Description

The set_output_delay, as well as the set_input_delay (see page 524) 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 (see page 518) 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 (see page 531) 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 (see page 517)

get_clocks (see page 514)

report_checks (see page 531)

set_clock_latency (see page 518)

set_output_delay (see page 528)

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 531). To exit interactive timer mode, call reset_sta (see page 534). 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 173: ***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, this command must be run again before interactive timing commands are used.

```
Command Syntax

prepare_sta (-slowc | -fastc) [-unconstrained]
```

Table 174: ***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.
`-fastc`	†	Use delays for the fast timing corner. This option is often used for verifying hold time requirements.
`-unconstrained`	✔	Enable reporting of unconstrained paths.

Table Note

† 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 175: *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.
-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.

Argument	Optional	Description
`-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`	✔	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.
`> 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 176: *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 -fast_c
```

ACE Tcl Commands

The following commands are used only within ACE. These are not available within Synplify, etc.

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>]	✔	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>]	✔	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>]	✔	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>]	✔	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 ../constraints/top.sdc:

```
add_project_constraints -project [get_active_project] "../constraints/top.sdc"
```

add_project_ip

```
add_project_ip <file> [-project <string>]
```

This command adds a link to an IP settings file to a project.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the IP settings file.
[-project <string>]	✔	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be added to.

add_project_netlist

`add_project_netlist <file> [-project <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>]`	✔	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be added to.

add_region_find_insts

`add_region_find_insts <region> <find_command> [-flops_only] [-clocks_only] [-include_constants] [-include_ascs_sacs] [-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]`	✔	When adding instances, filter out all instances except flops
`[-clocks_only]`	✔	When adding instances, filter out all instances with no connected clock
`[-include_constants]`	✔	When adding instances, do not filter out power/ground constants
`[-include_ascs_sacs]`	✔	When adding instances, do not filter out ASC/SAC instances
`[-batch]`	✔	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]`	✔	Print additional debug messages.

add_region_insts

`add_region_insts <region> <insts> [-flops_only] [-clocks_only] [-include_constants] [-include_ascs_sacs] [-batch] [-verbose]`

Add user design instances to a placement region constraint

Argument	Optional	Description
`<region>`		Name of the region
`<insts>`		List of user design instances
`[-flops_only]`	✓	When adding instances, filter out all instances except flops
`[-clocks_only]`	✓	When adding instances, filter out all instances with no connected clock
`[-include_constants]`	✓	When adding instances, do not filter out power/ground constants
`[-include_ascs_sacs]`	✓	When adding instances, do not filter out ASC/SAC instances
`[-batch]`	✓	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]`	✓	Print additional debug messages.

apply_highlights

`apply_highlights [-insts] [-nets] [-paths]`

This command updates the GUI with highlighting information on the present design.

Argument	Optional	Description
`[-insts]`	✓	Update highlighting of all instances in the design
`[-nets]`	✓	Update highlighting of all nets in the design
`[-paths]`	✓	Update highlighting of all paths in the design

apply_placement

`apply_placement [-defparams]`

Apply batch pre-placement commands

Argument	Optional	Description
[-defparams]	✓	The optional -defparams option specifies whether placement should be applied from defparams or from batch placement commands.

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.

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>]	✓	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>]	✓	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>]	✓	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>]	✓	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>]	✓	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>]	✓	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>]	✓	specifies name of domain

Argument	Optional	Description
`[-pin <string>]`	✓	use the domain of this pin
`[-net <string>]`	✓	use the domain of this net
`[-all]`	✓	even report uninteresting domains
`[-unique]`	✓	use a unique domain for domains with the same frequency
`[-multi]`	✓	with -net or -pin: report a list of domains instead of a single domain
`[-freq]`	✓	return the frequency (MHz); requires a domain
`[-period]`	✓	return the period (ps); requires a domain
`[-phase]`	✓	return the phase; requires a domain
`[-edge_type]`	✓	1 for pos-edge, -1 for neg-edge, 0 for combinational; requires a domain
`[-routing_props]`	✓	list of strings denoting the routing properties (if set); requires a domain
`[-core]`	✓	1 if used in the core, otherwise 0; or list of domains used in core
`[-driver]`	✓	name of the driving pin or port; requires a domain
`[-clock_net]`	✓	name of the clock net; requires a domain
`[-is_clock]`	✓	true if net is a clock net; requires a pin or net
`[-info]`	✓	list as for 'array set'; requires a domain
`[-group]`	✓	list of related domains, or list of groups
`[-equal]`	✓	list of domains with same frequency; requires a domain
`[-names]`	✓	list of all names for the domain; requires a domain
`[-sdc]`	✓	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]`	✔	apply current default_relation rule
`[-group]`	✔	values are in group units
`[-sdc]`	✔	return list of sdc commands

create_boundary_pins

```
create_boundary_pins <name> <boundary_pin_names> [-clock] [-data]
```

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]`	✔	Create a clock pin even if the specified net is a data net
`[-data]`	✔	Create a data pin even if the specified net is a clock net

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>]`	✓	The optional -command <command> option specifies the TCL command to run when this flow step is invoked.
`[-parent_id <string>]`	✓	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]`	✓	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]`	✓	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>]`	✓	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>]`	✓	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>]`	✓	The optional -project <projectName> option is used to specify an alternate project (by name) for the implementation to be added to.
`[-copy]`	✓	The optional -copy option is used to copy the implementation options of the active implementation into the newly created implementation.
`[-not_active]`	✓	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>]	✔	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>]	✔	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>]	✔	The optional -impl <implName> option is used to specify the name of the initial implementation for this new project.
[-not_active]	✔	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] [-soft] [-flops_only] [-clocks_only] [-include_constants] [-
include_ascs_sacs] [-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.
[-find_insts <string>]	✔	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.

Argument	Optional	Description
[-insts <list>]	✓	List of user design instances to constrain into this region's bounding box for the placer.
[-snap_to_clock_regions]	✓	Snap the bounding box to clock region boundaries
[-soft]	✓	Create "soft" placement region, which attempts to pull instance placement to its center, but allows instance placement to overflow the bounds of the region
[-flops_only]	✓	When adding instances, filter out all instances except flops
[-clocks_only]	✓	When adding instances, filter out all instances that do not have a connected clock pin
[-include_constants]	✓	When adding instances, do not filter out power/ground constants
[-include_ascs_sacs]	✓	When adding instances, do not filter out ASC/SAC instances
[-batch]	✓	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]	✓	Print additional debug messages.

deselect

```
deselect [-objects <list>]
```

This command removes objects from the current list of selected objects.

Argument	Optional	Description
[-objects <list>]	✓	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>]	✓	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to disable constraints for.
[-impl <string>]	✓	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>]	✓	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]	✓	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.
<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>]	✓	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>]	✓	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>]	✓	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]	✓	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>]	✓	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]	✓	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>]`	✔	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>]`	✔	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>]`	✔	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.
`[-batch]`	✔	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>]`	✔	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.

Argument	Optional	Description
[-layer <int>]	✔	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>]	✔	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>]	✔	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]	✔	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>]	✔	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]	✔	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>]	✔	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>]	✔	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>]	✔	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]	✔	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>]	✔	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]	✔	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>]	✓	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>]	✓	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>]	✓	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.
[-batch]	✓	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>]	✓	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]	✓	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.

Argument	Optional	Description
`[-layer <int>]`	✔	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>]`	✔	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>]`	✔	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]`	✔	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>]`	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to enable constraints for.
`[-impl <string>]`	✔	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]`	✔	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>]`	✔	Specifies the output file name for the partition database (default is <active_impl_dir>/output/partitions/<cellname>_<partition>.epdb)
`[-bboutputfile <string>]`	✔	Specifies the output file name for the partition blackbox Verilog model (default is <active_impl_dir>/output/blackboxes/<cellname>_bb.v)
`[-info_list]`	✔	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>]`	✔	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]`	✔	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]`	✔	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]`	✔	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]`	✔	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]`	✔	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]`	✔	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>]`	✔	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 ==.
`[-no_prefix]`	✔	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 385), Search Filter Builder Dialog (see page 285), Filter Properties (see page 329), find (see page 552), Search View. (see page 245)

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. Each pattern string in the list may use '*' and '?' wildcard characters for matching.
`[-insts]`	✓	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]`	✓	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]`	✓	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]`	✓	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]`	✓	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]`	✓	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>]`	✓	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>]`	✓	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>]`	✓	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".

Argument	Optional	Description
`[-no_prefix]`	✓	The optional -no_prefix option is used to remove the object type prefix from the names returned in the results
`[-no_refresh]`	✓	The optional -no_refresh option is used to prevent sending an update to the GUI Search View to optimize speed
`[-handle]`	✓	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]`	✓	Print warning message if the find command does not find any objects.
`[-error]`	✓	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 245). See also: Object Type Prefixes (see page 385), Search Filter Builder Dialog (see page 285), Filter Properties (see page 329), filter (see page 551), select (see page 592), Selection View (see page 249), trace_connections. (see page 598)

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>]`	✓	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]	✓	The optional -buildid option will return the buildid.
[-builddate]	✓	The optional -builddate option will say when ace was built.
[-full]	✓	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]	✔	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"  "AC16tSC01HI0
1C"
```

Also See

create_project (see page 543)

get_active_project (see page 556)

set_impl_option (see page 594)

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]	✔	do not print a message if there is no active project
[-path]	✔	Return the file path to the active project's acxprj project file, instead of the project name

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_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]`	✓	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]`	✓	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>]`	✓	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>]`	✓	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>]	✓	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>]
```

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>]	✓	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.

Argument	Optional	Description
[-impl <string>]	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.

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>]	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-impl <string>]	✔	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]`	✔	get timestamp
`[-comment]`	✔	get comment
`[-view]`	✔	get view name
`[-type]`	✔	get partition type

Argument	Optional	Description
[-is_import]	✔	was the partition imported (0 or 1)?
[-import_from]	✔	file the partition was imported from
[-changed]	✔	has the partition timestamp changed
[-id]	✔	unique partition id number
[-disabled]	✔	is the partition disabled (0 or 1)?
[-parent]	✔	name of the partition's parent partition
[-is_top]	✔	is this the top partition (0 or 1)?
[-is_parent]	✔	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]`	✔	The optional -pins option returns the list of pin names that make up this path.
`[-insts]`	✔	The optional -insts option returns the list of instance names on this path.
`[-nets]`	✔	The optional -nets option returns the list of net names on this path.
`[-frequency]`	✔	The optional -frequency option returns the frequency of this path in MHz. If no frequency is defined, -1 is returned.
`[-type]`	✔	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]`	✔	The optional -rgb option returns the integer rgb highlight color value for the path. A value of -1 means it is not highlighted.
`[-text]`	✔	The optional -text option returns the details text for this path.
`[-slack]`	✔	The optional -slack option returns the slack for this path.

get_placement

```
get_placement <objName> [-ball]
```

This command returns the site or ball name 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.
`[-ball]`	✔	The optional -ball option specifies whether the ball name of the specified object should be returned instead of the site name. This option is only valid if the object passed in is an IO buffer.

get_pod_names

```
get_pod_names [-all] [-usb] [-ethernet] [-list <list>]
```

Returns a list of names of available Bitporter pods.

Argument	Optional	Description
[-all]	✔	(default behavior) returns USB and detected Ethernet pods.
[-usb]	✔	(optional) returns only the USB pods.
[-ethernet]	✔	(optional) returns only the detected Ethernet pods.
[-list <list>]	✔	(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>]	✔	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>]	✔	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>]	✔	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>]	✓	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>
```

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.

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`

Returns the list of placement region constraint names

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]`	✔	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]`	✔	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]`	✔	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]`	✔	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]`	✔	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.
		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

Argument	Optional	Description
`[-paths]`	✔	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]`	✔	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]`	✔	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_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]
```

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>]	✓	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]	✓	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.

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

load_flowscripts

```
load_flowscripts
```

This command loads all of the encrypted flow scripts.

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]	✓	If this option is set, no impl in the project will be activated and the active impl in ACE will not be changed.
[-force]	✓	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]
```

This command prints a status message to the console and the log file.

Argument	Optional	Description
<msg>		The message to be printed.
[-info]	✓	Make this message an informational message.
[-warning]	✓	Make this message a warning message.
[-error]	✓	Make this message an error message.

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>]	✓	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>]	✓	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.

refresh_drawing

```
refresh_drawing
```

This command refreshes the current custom drawing on the GUI's Floorplanner view.

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 <implName> [-project <string>]
```

This command removes an implementation from a project. The implementation output directory on the file system is not deleted.

Argument	Optional	Description
<implName>		The required <implName> argument is used to specify the name of the implementation to remove.
[-project <string>]	✓	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>]	✓	Specifies the id of the path to remove
[-all]	✓	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>]`	✔	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_ip

```
remove_project_ip <file> [-project <string>]
```

This command removes the link to an IP settings file from a project. The IP settings file on disk is not deleted.

Argument	Optional	Description
`<file>`		The required <file> argument is used to specify the IP settings file (by file path).
`[-project <string>]`	✔	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be removed from.

remove_project_netlist

```
remove_project_netlist <files> [-project <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>]`	✔	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be removed from.

remove_region

```
remove_region [-region <string>] [-all]
```

This command removes a placement region constraint specification

Argument	Optional	Description
`[-region <string>]`	✔	Name of the region to delete
`[-all]`	✔	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>]	✔	List of user design instances to remove from this placement region constraint.
[-all]	✔	Clear all user design instances from this region constraint
[-flops_only]	✔	When removing instances, filter out all instances except flops
[-clocks_only]	✔	When removing instances, filter out all instances with no connected clock
[-verbose]	✔	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>]	✔	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>]	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to change the name for.

report_clocks

```
report_clocks
```

Report clocks in the current design

report_coverage

```
report_coverage [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>] [-verbose]
[-package]
```

Generate and write a coverage report for pins.

Argument	Optional	Description
[-outputfile <list>]	✔	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]	✔	The optional -text option is used to specify whether the file should be output as plain text.
[-csv]	✔	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	✔	The optional -html option is used to specify whether the file should be output as html-file.
[-columns <list>]	✔	The optional -columns option is used to specify a customized ordered list of columns names to output in the pin assignment report.
[-verbose]	✔	The optional -verbose option is used to specify whether the file should report ALL attributes and parameters for each IO port/pad.
[-package]	✔	Specify the -package option to output an additional full table with the mapping for the entire list of package balls

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>]	✔	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]	✔	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	✔	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	✔	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>] [-show_values] [-show_all]
```

Output a report of the current impl options defined in ACE

Argument	Optional	Description
`[-outputfile <string>]`	✔	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]`	✔	The optional -text option is used to specify that the file should be output as plain text.
`[-csv]`	✔	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>]`	✔	The optional -project <projectName> and -impl <implName> options can be used to specify an alternate project implementation (by name).
`[-impl <string>]`	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name).
`[-show_values]`	✔	If the -show_values flag is used, the report will output an additional column to display the current values for the given implementation. If no -project and -impl options are specified, the active impl will be reported.
`[-show_all]`	✔	If the -show_all flag is used, the report will additionally output all available non-standard options for the active impl. If this flag is not set, by default, all standard options that show in the GUI will be output.

report_partitions

```
report_partitions [-outputfile <list>] [-html] [-csv] [-text]
```

This commands generates and writes a formatted partition report

Argument	Optional	Description
`[-outputfile <list>]`	✔	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]`	✔	The -html option specifies that the output file(s) are written in HTML format (this is the default)
`[-csv]`	✔	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets

Argument	Optional	Description
[-text]	✔	The -text option specifies that the output file(s) are written in plain text format

report_pins

report_pins [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>] [-package]

Generate and write a pin to package assignment report

Argument	Optional	Description
[-outputfile <list>]	✔	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]	✔	The optional -text option is used to specify whether the file should be output as plain text.
[-csv]	✔	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	✔	The optional -html option is used to specify whether the file should be output as html-file.
[-columns <list>]	✔	The optional -columns option is used to specify a customized ordered list of columns names to output in the pin assignment report.
[-package]	✔	Specify the -package option to output an additional full table with the mapping for the entire list of package balls

report_placement

report_placement [-outputfile <list>] [-html] [-csv] [-text]

This command generates and writes a formatted placement QoR report

Argument	Optional	Description
[-outputfile <list>]	✔	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]	✔	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	✔	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets

Argument	Optional	Description
[-text]	✓	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]
```

This command generates and writes a formatted power dissipation report

Argument	Optional	Description
[-outputfile <list>]	✓	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]	✓	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	✓	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	✓	The -text option specifies that the output file(s) are written in plain text format
[-temperature <string>]	✓	Override the junction temperature printed in the report header
[-clocks <string>]	✓	The -clocks <{ {clk1 freq} {clk2 freq} .. {clkn freq} }> option may be used to specify the operating frequency 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]	✓	The -achieved option may be used to specify that achieved static timing results be used to calculate the power dissipation report for each clock

report_regions

```
report_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>]	✓	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.

Argument	Optional	Description
[-text]	✔	The -text option is used to specify whether the file should be output to the console as plain text
[-csv]	✔	The -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets

report_routing

`report_routing [-outputfile <list>] [-text] [-csv] [-html] [-nonterse] [-terse] [-overflowreportlimit <int>]`

This command generates and writes a formatted routing report.

Argument	Optional	Description
[-outputfile <list>]	✔	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]	✔	The optional -text option is used to specify whether the file should be output as plain text (Default is autodetect)
[-csv]	✔	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	✔	The optional -html option is used to specify whether the file should be output as html-file.
[-nonterse]	✔	normal information level.
[-terse]	✔	terse info level.
[-overflowreportlimit <int>]	✔	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>]	✔	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).

Argument	Optional	Description
[-html]	✓	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	✓	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	✓	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>]	✓	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]	✓	The optional -all option resets all impl options to their device-specific default values.
[-project <string>]	✓	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.
[-impl <string>]	✓	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>]	✓	Specifies an alternate project name to use instead of the active project when using the -impl <implname> option
[-impl <string>]	✓	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 – see Restoring Implementations (see page 359).

> **Restoring an Implementation clears current data**
>
> Restoring an Implementation (see page 312) 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 (see page 589) and Saving Implementations (see page 358).

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 last active impl from an ACXDB file (by default) to restore the state of the DB and impl options for a given impl. The -activeimpl option can be used to specify the impl name to activate and restore. By default, the .acxdb file is loaded from <project_dir>/<impl_dir>/<impl_name>.acxdb, unless the -acxdb option is used. 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 a project. If the -no_db option is used, the DB state of the active impl for the project will not be restored. 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]`	✔	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]`	✔	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]`	✔	If this option is set, the DB state of the active impl for the project will not be restored.
`[-activeimpl <string>]`	✔	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>]`	✔	Specifies an ACXDB file path from which to restore the state of the active impl.
`[-force]`	✔	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!

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>]`	✔	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>]`	✔	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]`	✔	The optional -resume option is used to run the entire flow from the last successfully completed flow step.
`[-ic <string>]`	✔	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>]`	✔	Clone Critical Instances that have slack lower than this limit
`[-fanout_limit <int>]`	✔	Apply fanout control on nets with fanout greater than this limit
`[-fanout_limit_clone <int>]`	✔	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>]	✔	Output directory name
[-download_pod_names <string>]	✔	(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>]	✔	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>]	✔	Output directory name
[-aeskey <string>]	✔	Key used for encryption. If not given, key is taken from impl. If not active in impl, the bitstream is not encrypted.

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>]	✔	The -debugdir <dir> option is used to override the default location for debug files during this step.
[-modelsdir <string>]	✔	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>]	✔	Output netlist file name.
[-final]	✔	Output DRC-free final netlist
[-compress]	✔	Compress output file with gzip

run_insert_holdbuffers

```
run_insert_holdbuffers [-margin <int>] [-io_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>]	✔	Insert a delay buffer per target pin whose hold time slack is less than the specified value
[-io_buffers <int>]	✔	Insert a delay buffer per target pin when driven directly by a flopped IO Pad/Pin

run_multiprocess

```
run_multiprocess [-use_existing_impls <string>] [-use_seeds <string>] [-
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 <string>] [-jobs_nfs_latency <int>]
```

This command runs the ACE multiprocess flow for the active implementation. By default, this command will generate new implementations from 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 <string>]	✔	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 [get_project_impls]
[-use_seeds <string>]	✔	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>]	✔	(optional) Sets the number of implementations to run in parallel.

Argument	Optional	Description
[-use_job_submission <int>]	✔	(optional) Set to a 0 to run jobs on the local machine, or set to a 1 to submit jobs to a cloud/grid/batch submission system.
[-stop_flow_at <string>]	✔	(optional) Specify the flow step ID string to stop the flow at.
[-copy_icdb <int>]	✔	(optional) Set to a 1 to copy the incremental flow DB from the template impl, or set to a 0 to not copy.
[-jobs_exec <string>]	✔	(optional) Specify the job submission system executable.
[-jobs_wd <string>]	✔	(optional) Specify the job submission system working directory argument.
[-jobs_name <string>]	✔	(optional) Specify the job submission system job name argument.
[-jobs_log <string>]	✔	(optional) Specify the job submission system job log argument.
[-jobs_args <string>]	✔	(optional) Specify the job submission system list of additional arguments, formatted as a list of TCL lists {{arg1 val1} {arg2 val2} ... }.
[-jobs_nfs_latency <int>]	✔	(optional) Specify the job submission system allowed seconds of NFS write latency.

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 371) section.

The GUI provides a graphical interface for multiprocess through the Multiprocess View (see page 204). See also: Running Multiple Flows in Parallel (see page 363), Attempting Likely Optimizations Using Option Sets (see page 434).

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>]	✔	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_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>]	✔	(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>]	✔	(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>]	✔	(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>]	✔	(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>]	✔	(optional) specifies the timeout in seconds before Snapshot is cancelled. If not specified, Snapshot will not timeout.

Argument	Optional	Description
[-verbose]	✓	(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>]	✓	(optional) specifies which JTAG pod connection to use. (If not specified, autodetection will be attempted.)
[-ir_bits_before <int>]	✓	(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>]	✓	(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>]	✓	(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>]	✓	(optional) specifies which recommended STAPL Procedures in the specified Action should be skipped.
[-enabled_procs <list>]	✓	(optional) specifies which optional STAPL Procedures in the specified Action should be executed.
[-defines <list>]	✓	(optional) list of definitions to add (e.g. 'rw_addr_int=4100').
[-log_file <string>]	✓	(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>]`

This command runs timing analysis on the design.

Argument	Optional	Description
[-prepared]	✔	Indicates that the design has only been prepared (this is the default)
[-placed]	✔	Indicates that the design has been placed but not routed
[-routed]	✔	Indicates that the design has been placed and routed
[-final]	✔	Indicates that this is sign-off timing (this involves some extra checks)
[-name_postfix <string>]	✔	Postfix added to report file name (e.g., to distinguish multiple 'placed' reports)
[-format <string>]	✔	Specify report formats; default is { text html csv }
[-temperature <string>]	✔	The temperature 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>]	✔	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>]	✔	The optional -reportsdir <dir> option is used to override the default location for report files during this step.

Argument	Optional	Description
`[-debugdir <string>]`	✔	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
`[-reroute]`	✔	The optional -reroute option is used to re-route the affected nets after un_post_process

run_unplace

`run_unplace [-fixed] [-boundary] [-io] [-core] [-constants] [-insts <list>]`

Unplace instances in the design

Argument	Optional	Description
`[-fixed]`	✔	Unplace instances with fixed placement constraints as well as movable instances
`[-boundary]`	✔	Only unplace boundary instances
`[-io]`	✔	Only unplace I/O-ring elements
`[-core]`	✔	Only unplace core elements
`[-constants]`	✔	Only unplace constant sources
`[-insts <list>]`	✔	Only unplace the instances specified in this list

run_unroute

`run_unroute [-net <string>] [-pin <string>] [-nets <list>] [-nocore] [-clock_only] [-core] [-keepsametile] [-uniqify] [-consts] [-keepconsts]`

Remove all or parts of a routing.

Argument	Optional	Description
`[-net <string>]`	✔	Only remove routing for given net
`[-pin <string>]`	✔	Only remove routing for given pin
`[-nets <list>]`	✔	route only the nets specified in this list
`[-nocore]`	✔	unroute only nets in the I/O-ring

Argument	Optional	Description
`[-clock_only]`	✔	only unroute clock nets
`[-core]`	✔	only unroute core nets
`[-keepsametile]`	✔	keep all same-tile - connections
`[-uniqify]`	✔	make constants unique again
`[-consts]`	✔	only unroute constants
`[-keepconsts]`	✔	do NOT unroute constants

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]`	✔	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 358) . See also: restore_impl (see page 579) and Restoring Implementations (see page 359).

save_placement

```
save_placement [-iofile <string>] [-corefile <string>] [-add] [-output_regions] [-
io_only] [-pads_only] [-core_only] [-fixed_only] [-fix] [-port_names] [-
device_port_names] [-ball_names]
```

Save the current placement to one or more files as a set of pre-placement commands

Argument	Optional	Description
`[-iofile <string>]`	✔	The -iofile <file> option is used to specify the file path to save the IO Ring 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.

Argument	Optional	Description
[-corefile <string>]	✓	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]	✓	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]	✓	The -output_regions option is used to enable output of region constraints into the Core PDC file (or IO Ring PDC file if -io_only is used)
[-io_only]	✓	The -io_only option is used to specify whether only the IO Ring pre-placement file is output or not
[-pads_only]	✓	The -pads_only option allows you to save only the placement of the io_buffer pad instances (or top-level ports if -port_names is used)
[-core_only]	✓	The -core_only option is used to specify whether only the Core pre-placement file is output or not
[-fixed_only]	✓	The -fixed_only option is used to specify whether only Fixed Instance placement data is output or not
[-fix]	✓	The -fix option forces all saved placements to be "fixed" placement, allowing you to lock down all of your placed instances
[-port_names]	✓	The -port_names option is used to specify whether port names should be output for IO Buffers instead of instance names
[-device_port_names]	✓	The -device_port_names option is used to specify whether device port names should be output for IO Buffers instead of site names
[-ball_names]	✓	The -ball_names option is used to specify whether ball names should be output for IO Buffers instead of site names

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>]	✔	The optional -project <projectName> option may be used to specify a project to write out.
[-outputfile <string>]	✔	The optional -outputfile <projectFile> option may be used to specify an output file location.
[-acxdb <string>]	✔	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]	✔	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] [-port_names]
```

This command is used to save all changed properties on objects in the DB after prepare has been run.

Argument	Optional	Description
[-outputfile <string>]	✔	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]	✔	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.
[-port_names]	✔	The optional -port_names option is used to specify whether port names should be output for IO Buffers instead of instance names.

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>]	✔	The name of the output file
[-region <string>]	✔	The name of the region to save. Saves explicit instance list.

Argument	Optional	Description
[-all]	✔	Save all regions (default)
[-explicit]	✔	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]	✔	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]	✔	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 249). See also: Object Type Prefixes (see page 385), Search View (see page 245), find (see page 552), Selecting Objects in the Floorplanner (see page 394), trace_connections. (see page 598)

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>]	✔	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> [-boundary] [-trunk] [-direct_trunk] [-fast] [-nominal] [-
low_jitter] [-local] [-global] [-immediate] [-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')
[-boundary]	✔	boundary clock (not routed through the trunk)
[-trunk]	✔	trunk clock
[-direct_trunk]	✔	routed from trunk to region without delay balancing
[-fast]	✔	use fast resources in the region
[-nominal]	✔	use nominal resources in the region
[-low_jitter]	✔	reduce jitter by avoiding dynamic clock muxes
[-local]	✔	use local (no qcm) route
[-global]	✔	use global (long with qcm) route
[-immediate]	✔	use local immediate (no qcm, no bcm) route
[-data_region]	✔	data as clock, using region resources
[-data_center]	✔	data as clock, using center resources
[-data_local]	✔	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>]	✔	Optional 2nd level cluster id

Argument	Optional	Description
[-wt <int>]	✔	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>]	✔	The new value to set the impl option to.
[-project <string>]	✔	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.
[-impl <string>]	✔	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

Argument	Optional	Description
`<name>`		Name of the Partition
`<changed>`		Set to 1 to mark this partition as force changed, set to 0 to mark as not forced changed

set_partition_info

```
set_partition_info [-name <string>] [-view <string>] [-cp_type <string>] [-cptype
<string>] -timestamp <string> -import <string> [-comment <string>]
```

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>]`	✔	Hierarchical Name of the Partition
`[-view <string>]`	✔	Module Name (View) of the Partition
`[-cp_type <string>]`	✔	Types are hard, locked, and soft
`[-cptype <string>]`	✔	TEMPORARY alias for cp_type
`-timestamp <string>`		Timestamp to indicate when this partition was last synthesized. Default is current time in milliseconds. The options -timestamp and -import are mutually exclusive
`-import <string>`		Adb filename to import partition from. The options -timestamp and -import are mutually exclusive
`[-comment <string>]`	✔	comment

set_placement

```
set_placement <objName> <siteName> [-fixed] [-batch] [-warning]
```

This command assigns the placement of an instance to a site

Argument	Optional	Description
`<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.
`<siteName>`		The required <siteName> argument is used to specify the site (s:) to place the instance on. In the case of IO pad placement, a ball (b:) name or 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]`	✓	The -fixed option specifies that the placement of the instance should be fixed to this site and not movable by the placer
`[-batch]`	✓	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.
`[-warning]`	✓	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

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>]`	✓	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>]`	✓	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.
`[-voltage <string>]`	✓	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]`	✓	The optional -warning option allows you to downgrade error messages about missing netlist objects to warning messages
`[-quiet]`	✓	The optional -quiet option allows you to disable printing of info messages

set_region_bounds

```
set_region_bounds <region> <bounds> [-snap_to_clock_regions]
```

This command updates a placement regions 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]`	✓	Snap the bounding box to clock region boundaries

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]
```

source encrypted tcl file.

Argument	Optional	Description
`<tclfile>`		encrypted tcl file to source
`[-nodigest]`	✓	no digest is also OK

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]`	✓	If you want to select only upstream logic that drives the instances in the current selection, use -drivers_only
`[-targets_only]`	✓	If you want to select only downstream logic driven by the currently selected instances, use -targets_only.
`[-include_clocks]`	✓	To include tracing connectivity on clock nets, use -include_clocks.
`[-include_resets]`	✓	To include tracing connectivity on reset nets, use -include_resets.

Similar to the find (see page 552) command, this functionality is especially useful when creating lists as input to other Tcl commands, like select (see page 592) and highlight (see page 568).

write_bitstream

```
write_bitstream [-outputfile <string>] [-debugdir <string>] [-reportsdir <string>] [-
jam] [-flash] [-flash4x] [-hex] [-cpu] [-cpu_width <int>] [-flash_clock_div <int>] [-
nocompress] [-compress] [-chainfile <string>] [-aeskey <string>]
```

This command generates a programming bitstream for a fully placed and routed design in STAPL format.

Argument	Optional	Description
`[-outputfile <string>]`	✓	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.

Argument	Optional	Description
[-debugdir <string>]	✔	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
[-reportsdir <string>]	✔	The optional -reportsdir <dir> option is used to override the default location for report files during this step.
[-jam]	✔	The optional -jam option may be used to output an additional jam-file
[-flash]	✔	The optional -flash option may be used to output an additional serial flash binary file format output
[-flash4x]	✔	The optional -flash4x option may be used to output 4 additional 4x serial flash binary file format outputs
[-hex]	✔	The optional -hex option may be used to output an additional raw hex file format output
[-cpu]	✔	The optional -cpu option may be used to output an additional CPU Mode file format output
[-cpu_width <int>]	✔	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>]	✔	This option specifies the Serial Flash clock divider value to be used when programming the chip from Serial Flash
[-nocompress]	✔	write output as plain-text file
[-compress]	✔	compress output-file
[-chainfile <string>]	✔	The optional -chainfile <file> may be used to override the chainfile set in the active Impl
[-aeskey <string>]	✔	Key used for encryption. If not given, key is taken from impl. If not active in impl, the bitstream is not encrypted.

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>]	✔	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>]	✔	Output netlist file name.
[-debugdir <string>]	✔	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
[-final]	✔	Output DRC-free final netlist
[-compress]	✔	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>]	✔	Specifies the name of the blackbox model library (default is "blackbox")
[-outputfile <string>]	✔	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>]`

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

Argument	Optional	Description
`[-outputfile <string>]`	✓	Specifies the output file name for the partition database (default is partitions/<instname>.epdb in the implementation output directory)

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 will likely be able to 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 may 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 344) 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 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.	Linux users should unset the LD_LIBRARY_PATH environment variable and try running ACE again. If problems persist, contact Achronix Technical Support.
100 / 101	The GUI is attempting to workaround a number of known startup issues through automated forced restarts (which display these exit codes). If the restarts failed 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 the user attempting to execute ACE in an unsupported OS.)
404	The user attempted to run ACE 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)
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. Customers seeing this limit should contact their FAE for current workarounds.

> 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 username includes German extended characters (e.g. umlauts), Chinese, etc, it may cause ACE to 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 313)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.

> Do not use this forced unlock procedure to run multiple ACE sessions on the same project simultaneously, or file corruption may occur!
>
> Project definitions, Implementation definitions, saved implementation states (for both normal flow and incremental compilation), log files, and output directories may 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 may 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 258), 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 may 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 will force ACE to load (or restore) the project despite the presence of a lock, and this session of ACE will re-lock 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

The font used in most ACE Views is directly inherited from the underlying GUI application framework stack, called the "Application Font" (Linux/GTK2), and "Message Box" font (Windows). The views within ACE will often accept font changes immediately, but in some cases users might need to restart ACE to see the font changes propagate completely throughout the application.

> It is highly recommended that users choose a plain font (not bold, not italics)!

Linux:

The config location can vary in every Linux desktop / version / distro. In Linux, ACE uses the GTK+ widgets and fonts, so those are the settings users should change.

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**.

Windows:

In Windows 7, the Font configuration can be found under **Control Panel** → **Appearance and Personalization** → **Personalization** → **Window Color** → **Advanced appearance settings...** → **Item:Message Box** (then pick the desired font and size).

Fonts in HTML Reports

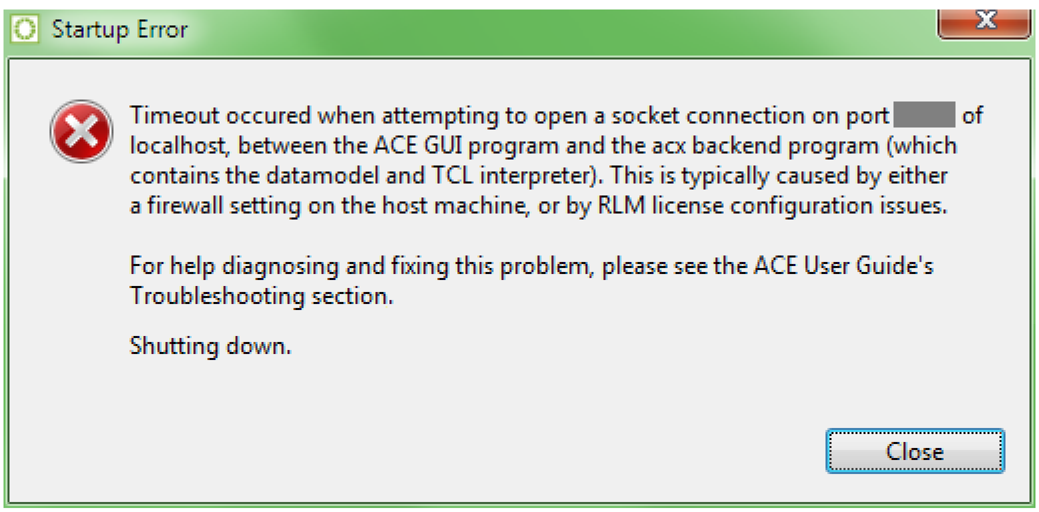
The font used in the HTML Reports is directly inherited from the system's HTML browser's font settings. The system HTML browser can be Internet Explorer (Windows) or WebKitGTK (Linux). The HTML reports will typically use the "Proportional" (sometimes called "Web Page") font and Monospace (also called "Plain Text") font types -- change these settings in the system's HTML browser to make the fonts change in the ACE HTML Report viewer.

Windows

In Windows 7, the browser font configuration is found under **Control Panel** → **Network and Internet** → **Internet Options** → **Internet Properties** → **General** → **Appearance** → **Fonts**. There, users can choose both the proportional "Webpage font" and the fixed-width "Plain text font".

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 Whether ACE is Experiencing a Firewall or Licensing Problem

Attempting to run ACE in batch mode (with `ace -batch`, see Running ACE (see page 344)) 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 a licensing configuration problem.

If batch mode ACE starts successfully in less than 60 seconds, then there is a firewall configuration problem.

> ⚠ Though unlikely, realize it is possible for both firewall and licensing problems to occur for the same user. Users might need to first fix a firewall problem, and then fix a licensing problem. After attempting one kind of fix, if the dialog continues to appear, users should 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.

Once batch mode is 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 344) 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 may need special permissions to be added to the firewall of the workstation running ACE. Users may need help from their local system administrator and /or network administrator to properly configure their firewall to allow the necessary network traffic.

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 may 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). Users must ask their network or system administrator exactly which port number should be specified - this will be 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 may need to be opened specifically for the licensing software. Refer to the *ACE Installation and Licensing Guide* (UG002) for details.

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 may 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. Incorrect numeric values cause a warning to be logged, but are ignored. Please work with your IT department for help setting the environment variable.

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's a user preference on the Multiprocess: Configure Custom Job Submission Tool Preference Page (see page 302) called **Allowed seconds of NFS write latency**. To fix the problem, increase the value of this preference to allow more time between when a the implementation's flow process has completed 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 may 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 user first changes to the Floorplanner Perspective, the ACE GUI window may become solid grey, and the title bar may 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 may 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 the ACE GUI 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 296), (**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.

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, users should 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 may 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 the user starts ACE, to have the user 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. Users can query all their current ulimit settings using '`ulimit -a`', or can 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`'. Lucky users might be able to remove the limit entirely with '`ulimit -n unlimited`'. (Again, users of other shells will need to use a different command.)

It is highly recommended that these file limit changes be done under the supervision of the system administrator. System administrators will often apply upper bounds for such ulimit assignments, and individual users won't be able to exceed those upper bounds without system administrator assistance.

If raising the open files limit does then 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, the user could create a script used to start ACE. In that script, the file limit could be temporarily raised before starting ACE, and then lowered again once 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 user moves and resizes the window), the ACE window will be 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 by users, especially if users already have a minimized application icon in that region of their screen.

Once ACE is running in that tiny window, twm users can move and enlarge the ACE window in the same manner they would 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 will remember the position and dimensions at application shutdown, and the next time the application is started, ACE will return 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 (CentOS5 can talk to CentOS6, but CentOS5 should not talk to CentOS7).

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 6 but having their X DISPLAY redirected to a CentOS 4 workstation, or vice-versa. Users attempting to bridge multiple OS revisions in this way will 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 6 and display on RHEL4.

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 the user's 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 their 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 will crash 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.

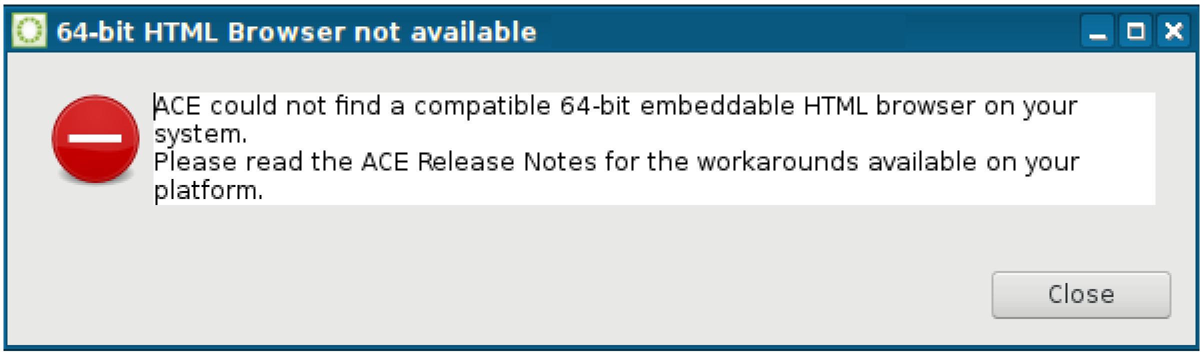
Achronix does not support ACE when forced to run using LD_LIBRARY_PATH.
At their own risk, users may add the command-line argument "`-enable_ld_library_path`" to force ACE to keep the preassigned LD_LIBARY_PATH value at startup.

```
./ace -enable_ld_library_path
```

Linux: Incompatible Default Web Browser

Caution!

The following content is not relevant to ACE releases prior to v7.1, which all lacked support for GTK+3. The following content is only relevant to releases that support GTK+3, meaning ACE v7.1 and later.



At startup, ACE tries to find a compatible 64-bit embeddable HTML browser already installed on the system, so that ACE can be used 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).

Starting with ACE v7.0, ACE now uses WebKitGTK (instead of Mozilla XulRunner) to display HTML web pages within ACE. This is used for both the help system and HTML report browsing.

Starting with ACE v7.1, ACE added support for GTK+3, and is thus able to use WebKit2GTK as well (WebKit2GTK is the successor to WebKitGTK).

Mozilla XulRunner (used in earlier versions of ACE) was replaced by WebKitGTK because XulRunner is not compatible with GTK+3, and is thus is no longer reliably found within modern Linux distributions. In contrast, multiple versions of WebKitGTK are typically available in every Linux distribution.

RHEL/CentOS v6.x and v7.x customers are not expected to experience any problems with web browser support. Customers running on unsupported Linux distros may 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, the user should work with their IT department to install an ACE-compatible 64-bit web browser in their distribution.

For RHEL/CentOS6 (and other GTK+2 Linux distros using the "classic" GUI interface), ACE-compatible 64-bit web browser libraries include WebKitGTK+ 1.2.x and newer, or 2.x and newer, but only if compiled for GTK+2 support (this typically excludes all versions of WebKit2GTK).

For RHEL/CentOS7 (and other GTK+3 Linux distros using the "modern" GUI interface), any WebKitGTK or 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.

Details

GTK+2

For GTK+2 Linux distros (like RHEL/CentOS6), with the assistance of a system administrator, users will need to install a GTK+2 compatible version of WebKitGTK. This is expected to be a package named "webkitgtk.x86_64", currently with a version of 2.4.9 or later, though any version after 1.2.x is expected to work, as long as it is compatible with GTK+2. The needed package is found within the official RHEL/CentOS6 distribution repositories and is typically installed by default.

> **Note**
>
> There are no known available packages for WebKit2 compatible with GTK+2.

GTK+3

For GTK+3 Linux distros (like RHEL/CentOS7), with the assistance of a system administrator, users will need to install a GTK+3 compatible version of WebKitGTK ***or*** WebKit2GTK. In RHEL/CentOS, these are expected to be either of the following:

- (preferred) a package named "webkitgtk4.x86_64", which contains WebKit2 compiled for GTK+3
- (also acceptable) a package named "webkitgtk3.x86_64", which contains WebKit compiled for GTK+3; recommended is a version of 2.4.9 or later, though any version after 1.2.x should work, as long as it is compatible with GTK+3

The needed packages are found within the official RHEL/CentOS7 distribution repositories and are often installed by default.

Additional Information

RHEL/CentOS7

RHEL/CentOS7 includes versions of WebKitGTK+3 (the package "webkit3.x86_64") and WebKit2GTK+3 (the package "webkit4.x86_64") within the official release repositories. Both of these are compatible with ACE when running on RHEL /CentOS7. Both of these packages are often installed by default.

> **When using GTK3, if browser problems are experienced, it might be necessary to force WebKit2 to be disabled**
>
> WebKit2 support is only available in combination with GTK+3 support, and thus requires the "modern" UI framework available in ACE v7.1 and later.
>
> On GTK+3, ACE will use WebKit2 by default when found. If problems are experienced, users might find it necessary to force ACE to use the older, more stable WebKit(1) instead, by disabling WebKit2 support.
>
> Setting the system environment variable "`SWT_WEBKIT2=0`" before running ace will disallow WebKit2 and thus force ACE to use the older WebKit(1) support. With GTK+2, requiring the package "webkitgtk.x86_64" (on RHEL /CentOS6), while with GTK+3, requires the package "webkitgtk3.x86_64" (on RHEL/CentOS7).
>
> Setting the system environment variable "`SWT_WEBKIT2=1`" before running ace enables WebKit2 support in GTK+3, if the package "webkitgtk4.x86_64" is found (in RHEL/CentOS7). This is the default mode starting in ACE v7.1. However, this setting does nothing useful when ACE is running with GTK+2, since WebKit2 is typically incompatible with GTK+2, and thus only the older WebKit(1) will be used with GTK+2.

RHEL/CentOS6

ACE is fully compatible with RHEL/CentOS6 using the inbuilt WebKitGTK+2 package (named "webkitgtk.x86_64").

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/CentOS. (Some other Linux distributions 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 the official RHEL/CentOS6 repositories; found in non-official community repositories for RHEL/CentOS7)
- ~~`webkitgtk2`~~: WebKit2 compiled for GTK+2? (not available for RHEL/CentOS; this appears to only have been an option during the very early development of WebKit2?)
- `webkitgtk3`: WebKit(1) compiled for GTK+3 (incompatible with CentOS6; found in the official RHEL/CentOS7 repositories)
- `webkitgtk4`: WebKit2 compiled for GTK+3 (incompatible with CentOS6; found in the official RHEL/CentOS7 repositories; this is the WebKitGTK package most actively developed/supported for RHEL/CentOS)

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 pkgs.org. At that site, search for "webkitgtk" in your chosen Linux distribution, find one that will work with the required version of GTK+, and then install it with the local package management tool.

The "classic" and "modern" Eclipse UI Frameworks

Achronix uses two versions of the Eclipse framework to build the ACE application for Linux GTK. At startup, ACE looks up which RHEL or CentOS distro version is running and chooses the appropriate version of the Eclipse framework to be used for the duration of the session:

- For RHEL and CentOS 7 and later, ACE uses a more 'modern' version of the Eclipse framework, v4.10 (codename 2018-12). This is the UI framework with GTK+3 support.
- For RHEL and CentOS 6, and for improved backwards compatibility with (older) unsupported Linux distros, ACE also ships a 'classic' version of the Eclipse framework, v4.6 (codename Neon). This is the UI framework with GTK+2 support.

- If ACE is unable to determine the RHEL/CentOS version (as would happen on unsupported distros), ACE defaults to using the 'modern' Eclipse v4.10 (codename 2018-12) framework.

The GTK+2 compatible classic v4.6 (Neon) version of the Eclipse framework may be forced by starting the ace launcher with the `'-force_classic_ui'` command-line argument. This framework version is required for RHEL/CentOS 6 since the OS only supports GTK+2. This classic version may also run acceptably on RHEL/CentOS 7 and later if GTK+2 (and a related version of WebKitGTK) is installed.

> **Note**
>
> This classic framework version ***requires*** GTK+2 v2.18 or later.

The more modern v4.10 (2018-12) version of the Eclipse framework, which is incompatible with GTK+2, may be forced by starting the ace launcher with the `'-force_modern_ui'` command-line argument. This framework version ***requires*** GTK+3 v3.8 or later, which is why it will not work with RHEL/CentOS 6. This more modern version is strongly recommended on all desktop environments compatible with GTK+3.

For the complete (and painful) details on getting the Eclipse v4.6 (Neon) and v4.10 (2018-12) frameworks working (with various versions of web browsers, GTK, Cairo, Wayland, etc.) on unsupported Linux distros, refer to the Eclipse SWT FAQ at https://www.eclipse.org/swt/faq.php

Linux: GTK+3: ACE Requires an Unusually Large Amount of Virtual Memory (Due to WebKit2)

> **Note**
>
> Linux users running on GTK+2 (for example, on RHEL6 or CentOS6) can ignore this section. WebKit2 is not available on GTK+2, thus it will not affect GTK+2 users (or their virtual memory allocations).

When running on GTK+3 versions of Linux (such as RHEL7 or CentOS7), the Eclipse framework underlying ACE defaults to using the WebKit2 HTML browser framework. This framework uses large amounts of virtual memory (sometimes more than 100GB), apparently as part of the browser's security model. At the present time, there appears to be no way (through the Eclipse framework) for Achronix to ask/force WebKit2 to use less virtual memory in ACE.

If these large amounts of virtual memory are causing problems, it is possible to make the Eclipse framework (and thus ACE) use the older WebKit(1) HTML browser instead of WebKit2. The older WebKit(1) browser framework does not require large amounts of virtual memory.

To force the Eclipse framework underlying ACE to use WebKit(1) instead of WebKit2, before starting ACE, set the system environment variable "SWT_WEBKIT2" to "0", using "export" or "setenv" (as appropriate to your choice of shells).

```
In bash:
$ export SWT_WEBKIT2=0
$ ./ace

In csh/tcsh:
$ setenv SWT_WEBKIT2 0
$ ./ace
```

If WebKit2 has already been disabled in this manner, and the user wishes to re-enable it, simply set the environment variable "SWT_WEBKIT2" back to "1", and restart ACE.

Linux: ACE Draws Slowly Onscreen (or Looks Ugly); can I Change This?

Themes

ACE currently uses the GTK widgets library in Linux, which supports 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 will vary) setting in Linux.

Note: Achronix does not recommend the use of the theme 'Oxygen-gtk' with ACE.

The 'Oxygen-gtk' theme engine from KDE is known to cause many errors with the Eclipse framework underlying ACE. Because of the instability, and because it is not supported by GTK themselves, the Eclipse framework team strongly recommends against its use. Because ACE uses the Eclipse frameworks, ACE is unable to support running on the 'Oxygen-gtk' theme.

Customers choosing to run ACE on systems with KDE installed should select a different GTK+ widget theme, not 'Oxygen-gtk'.

"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 will vary by Linux distribution and version, as well as by desktop manager and version.

Under CentOS6 Gnome, for example:
System → **Preferences** → **Appearance** → **Theme**

Under CentOS6 KDE, for example:
at the command prompt, start "`gnome-appearance-properties`", which opens the GTK+ Theme chooser

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**

Users should consult with their local System Administrator for additional details regarding the configuration of GTK+2 and /or GTK+3 for their local Linux installation.

Themes and GTK2 (RHEL/CentOS6)

Changing themes will not only alter the appearance of ACE, but it can also change the rendering speed of the widgets themselves. GTK+2 themes appearing simple/flat/square (like 'Mist', 'Simple', and 'ThinIce') can be faster to render than complex/rounded themes (like 'GrandCanyon' or 'eXperience'). At some customer sites (like those with poor bandwidth or high network latency between the X server and the X client), the choice of themes may have a significant impact upon the perceived ACE render performance. For these sites with serious render latency issues, customers might find it useful to test/compare the performance of their installed Themes using one of the free open-source tools available on the internet.

Themes and GTK3 (RHEL/CentOS7)

> **The default GTK+3 theme 'Adwaita' is the only theme supported by ACE when running on GTK+3**
>
> The Eclipse GUI framework underlying ACE only guarantees correct behavior and appearance on GTK+3 when the Adwaita GTK+3 theme is in use. Thus ACE is only officially supported running on the Adwaita GTK+3 theme. Because Adwaita is the default GTK+3 theme in RHEL/CentOS7, 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.

For customers desiring to use an alternate (non-Adwaita) theme on GTK+3, Achronix recommends that GTK+ v3.22 or later (available as of RHEL/CentOS 7.4) be used with ACE if possible, because those GTK+ versions are reportedly the "final, stable" release of GTK+3, and are thus the most stable with other themes. 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.

Again, if the version of GTK+3 at a customer site is a release prior to v3.22, then only the 'Adwaita' GTK theme can be recommended by Achronix for best results with ACE (and all other tools built upon the Eclipse frameworks).

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 Adwaita theme is not available.

Themes for RHEL/CentOS 7.2 and later (Linux using GTK3.14+)

When running on GTK3.14+, ACE enforces the usage of the Adwaita theme for the ACE application itself, regardless of what theme is chosen by the OS or desktop environment. This is done to maximize application performance and stability.

Customers wishing to use alternate (non-Adwaita) themes do so at their own risk – stability issues and render bugs are likely to occur, according to the Eclipse framework support forums.

By starting ACE with the command-line argument '`-keep_user_gtk_theme`', ACE will stop enforcing the usage of the Adwaita theme.

Themes for RHEL/CentOS 7.0 and 7.1 (Linux using GTK3 prior to version 3.14)

ACE is unable to enforce the use of the Adwaita theme with older versions of GTK3. Users noticing painting or stability problems in the GUI may need to manually choose the "Adwaita" theme at the system level. See the previous section (see page 615) about picking a GTK Application Widgets Theme for more details.

Animations and Other Effects

While desktop, application/window, and widget animations can improve the feel of applications for some users, other users will want to avoid the negative performance impacts.

Within ACE itself, a few supplemental animations can be enabled and disabled through an ACE preference setting. This is found as a checkbox at:

Window → Preferences → General → Appearance → Enable Animations

Additional 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 CentOS6 Gnome, for example, desktop effects can be found at:

System → Preferences → Desktop Effects

Under CentOS6 KDE, for example, animation and other special effects can be found at:

System Settings → Desktop Effects

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

Users should consult with their local System Administrator for additional details regarding the configuration of desktop, application, and widget animations and special effects for their 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.)

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. We have not yet heard any reports of the bug being observed when the system default GTK themes (System on CentOS6 and Adwaita on CentOS7) were in use.

See the previous section titled Themes (see page 615) 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

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 may 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.

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.

Upgrading an ACE Installation

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. Uninstall the prior version of ACE
3. Install the desired version of ACE
4. Re-connect any USB Bitporters
5. Run ACE

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 will also be the first one in the PATH environment variable, which will affect which version of ACE and `acx_stapl_player` 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 will ensure 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 will also make 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 user mistakenly installs a different version of ACE on top of an existing prior installation,) ACE GUI errors or crashes may occur, especially in the IP Editors.

If the user doesn't wish to perform an uninstall/reinstall of ACE, the following steps will often solve the problem:

1. Delete the "`.eclipse`" subdirectory (the leading '.' is important!) in the user's home directory.
   - (Windows:) typically "`C:\Users\username\.eclipse\`"
   - (Linux:) typically "`/home/username/.eclipse/`"

   Caution!

   This subdirectory is hidden in Linux; if you don't know 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 you 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 the user's home directory.
3. Run ACE

Revision History

Version	Date	Description
7.0	December 7, 2018	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). Concepts: • The Floorplanner View (see page 178) has changed the presentation of routing errors (open connections, open pins, and route overflows) • The Speedster22i Basic PLL Configuration Editor (see page 43)'s Basic PLL Overview Page (see page 44) includes new feedback to show the difference between the requested frequency and achieved frequency • Updated the Critical Path Diagram View (see page 170) now that it has similar right-click context menu items to the other views within the Floorplanner Perspective • Updated the Options View (see page 215) section 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 325) • Deleted the now-obsolete page "Fast Views". Tasks: • Added several pages describing Accessing Help (see page 502). • Added a new page describing Multiprocess Batch Mode (see page 371) (using the new run_multiprocess (see page 583) Tcl command), including the new seed sweep functionality. • Added a new page about Detaching Views and Editors (see page 347). • The Moving and Docking Views and Editors (see page 346) page and Rearranging Tabbed Views and Editors (see page 346) page have been re-titled and have had their content updated to reflect that in addition to Views, the Editors can now be moved around too. Additionally, the user feedback has changed during movement and docking, and the descriptions/tables have been updated accordingly. • The now-obsolete page "Opening Perspectives" page has been deleted. ACE Tcl Commands: • Added new command run_multiprocess (see page 583) • Added "-verbose" option to: add_region_find_insts (see page 536), add_region_insts (see page 536), create_region (see page 543), remove_region_insts (see page 573) • Added "-warning" and "-error" options to find (see page 552) • Added "-modelsdir" option to run_generate_fullchip_sim (see page 582)

Version	Date	Description
		• added "-clock_only" 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 573) • added "-flops_only" option to remove_region_insts (see page 573) • added "-io_buffers" option to run_insert_holdbuffers (see page 583) • set_false_path (see page 523) has improved description of various options Troubleshooting (see page 602): • Linux: 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. • Linux: added a section describing the known (GTK theme-dependent) bug where views/editors may detach (instead of move) when dragged while the Help Window is open. • Linux: added a section about the impacts of Linux resource limits. • Linux: added a section about the twm Window Manager in Linux. • Linux: added a section regarding Linux LD_LIBRARY_PATH concerns. • Linux: added a section about dialogs in the CDE Window Manager. • Linux: the web browser section has been updated to reflect ACE's migration from the Mozilla XulRunner to the WebKitGTK+ framework.
7.1	March 27, 2019	Advanced Concepts: • Added a new page describing ECO Commands (see page 332). Concepts: • Updated information on the Implementation Options Report (see page 325) page to improve clarity and accuracy. • Added info to the Properties View (see page 242) page covering double-click shortcut gestures and context-menu actions. • Updated Projects View (see page 238) page with latest actions Tasks: • Clarified the content of the Multiprocess Batch Mode (see page 371) page Troubleshooting: • Linux: Updated the support information for web browsers and themes to reflect how ACE now supports GTK3 and WebKit2.
		Tcl Reference: • updated the descriptions for all SDC Commands (see page 510), 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 529). Views:

Version	Date	Description
7.2	June 6, 2019	• updated the Flow View (see page 187) page, updating the screen shot and icon images to match recent updates • updated the Multiprocess view (see page 204) page, updating the screen shot and the text to include the new Seed Sweep functionality • updated the Netlist Browser view (see page 211) actions table, adding missing actions and updating icons • updated the Package View (see page 226) and Floorplanner View (see page 178) to mention the new **Allow Tooltips** toggle checkbox. • updated the Critical Paths View (see page 174) to mention the new Action to **Show Clock Paths**. Advanced Concepts • updated the ECO Commands (see page 332) section, moving each dialog into its own page and updating screen shots Troubleshooting 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).