



Libero IDE v5.0
User's Guide

Actel Corporation, Mountain View, CA 94043-4655

© 2003 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 5029120-5

Release: August 2003

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.

Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.

This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

Trademarks

Actel and the Actel logotype are registered trademarks of Actel Corporation.

Adobe and Acrobat Reader are registered trademarks of Adobe Systems, Inc.

Liberty is a licensed trademark of Synopsys Inc. This product uses SDC, a Proprietary format of Synopsys Inc.

Libero is a trademark of Actel Corporation.

Mentor Graphics, Viewlogic, ViewDraw, MOTIVE, and Model*Sim* are registered trademarks of Mentor Graphics, Inc.

Synplify and Synplicity are registered trademarks of Synplicity, Inc.

Verilog is a registered trademark of Open Verilog International.

WaveFormer Lite and SynaptiCAD are trademarks of SynaptiCAD, Inc.

Windows is a registered trademark and Windows NT is a trademark of Microsoft Corporation in the U.S. and other countries.

All other products or brand names mentioned are trademarks or registered trademarks of their respective holders.

Table of Contents

1. Introduction	1
2. Project Management	11
3. HDL Entry	33
4. Schematic Entry	37
5. Design Constraints	43
6. Design Implementation	153
7. Tcl Scripting	200
8. SmartPower	273
9. Timing Analysis	301
10. Using Timer	312
11. Synthesis	345
12. Testbench Creation	351
13. Simulation	355
14. Device Programming	363
15. Contacting Actel	371

Introduction

Welcome to Libero IDE

Libero IDE is the most comprehensive and powerful FPGA design and development software available, providing start to finish design flow guidance and support for novice and experienced users. Libero IDE combines Actel tools with such EDA powerhouses as Synplify, ModelSim, ViewDraw, WaveFormer Lite, and Silicon Explorer.

Libero IDE software supports all Actel Flash and antifuse products, including the popular ProASIC^{PLUS} and Axcelerator products.

Getting Started Quickly with Libero IDE

The Libero IDE design flow consists of the following steps.

- Create a new project or open a project
- Create source files (schematic, HDL, and ACTgen Macros)
- Import source files into your project
- Synthesize your design with Synplify (non-schematic designs)
- Perform Functional Simulation with *ModelSim*
- Implement your design with Actel's Designer software
- Perform Timing Simulation with *ModelSim*
- Program your device
- Debug your device

What's new in Libero IDE v5.0

Choose which tools to integrate

Using the project profile, specify which tools to integrate with your project. You can also set profiles for each tool, specifying the tool name, location, and other parameters.

Create new projects quickly

The New Project Wizard helps you create new projects quickly and easily. Select the project tools and import source files to get your started fast.

Design flow guidance

The Process window guides you through the design process, indicating completed steps and where you are in your design flow.

Operate more synthesis tools

Libero IDE now supports Synplify™ from Synplicity, LeonardoSpectrum™ from Mentor Graphics®, and Precision RTL™ from Mentor Graphics.

Choose your text editor

Use the Libero HDL Editor or setup your own preferred text editor.

Search and open files

The powerful new search feature finds files and posts results in the log window. Simply click a search result to open the file.

More Save options

Use the Save As command to create a copy of your project or to change the project name.

Specify VHDL Package compilation order

Specify the sequence in which your VHDL Package files should be compiled.

User defined “Do” file

For simulation, you may now use a “Do” file that is created outside of Libero IDE, or use the file automatically created by Libero IDE. It is also possible to include a Do file with the automatically created Do file.

Unknown hierarchy

The File Manager now indicates files that cannot be fully read by Libero.

FlashPro programmer support

With the integration of FlashPro, Libero now takes you from design creation to device programming., by invoking FlashPro and passing the project programming files to it.

ChipPlanner (Designer)

ChipPlanner is a graphical applications for viewing and placing I/O and logic macros. You can also use it for floorplanning. This tool is particularly useful when you need maximum control over your design placement in order to achieve optimum trade-offs between chip density and device performance.

Libero Editions

A description of the different Libero editions can be found online at <http://www.actel.com/products/tools/libero/libversions.html>.

Minimum System Requirements:

- 300MHz Pentium or greater
- 128 MB RAM
- FAT32 or NTF (NTFS file system strongly recommended)
- 600 MB available hard disk space minimum, 1 GB for full install of Libero IDE and all libraries
- 100 MB free space in C: drive for installation file swapping is recommended
- 5 MB available hard disk space for each VITAL/VHDL device family simulation library
- 5 MB available hard disk space for each Verilog device family simulation library
- CD-ROM drive
- HTML browser
- 800x600 video resolution

Libero Tools

The Libero Project Manager integrates design tools, streamlines your design flow, manages design and log files, and passes design data between tools.

Function	Tool	Company
Design Entry (HDL)	HDL Editor	Actel
Design Entry (Schematic)	ViewDraw AE	Actel
Synthesis	Synplify	Synplicity
Synthesis	Precision	Mentor Graphics
Synthesis	LeonardoSpectrum	Mentor Graphics
Simulation	ModelSim	Mentor Graphics
Automatic Testbench Generator	WaveFormer Lite	SynaptiCAD
Automatic Macro Generator	ACTgen	Actel
Place-and-Route	Designer	Actel
Programming Software	FlashPro	Actel
Programming Software	APS/Silicon Sculptor	Actel
In-Silicon Debug	Silicon Explorer II	Actel

HDL Editor targets the creation of HDL code. HDL Editor supports VHDL and Verilog with color, high-lighting keywords for both HDL languages.

ViewDraw AE is the Libero schematic entry vehicle.

Synplify AE from Synplicity, is integrated as part of the design package, enabling designers to target HDL code to specific Actel devices.

Leonardo Spectrum from Mentor Graphics can be integrated with Libero, but it does not come as part of the Libero tool set.

Precision from Mentor Graphics can be integrated with Libero, but it does not come as part of the Libero tool set.

Actel Designer software package includes:

- PinEditor package level floorplanner and I/O attribute editor
- ChipEditor chip level module placer
- ChipPlanner for floorplanning
- NetlistViewer design schematic viewer
- SmartPower power analysis tool
- Timer static timing analysis and constraints editor
- ACTgen macro generator

ModelSim from Mentor Graphics enables source level verification so designers can verify HDL code line by line. Designers can perform simulation at all levels: behavioral (or pre-synthesis), structural (or post-synthesis), and back-annotated, dynamic simulation. (ModelSim is supported in Libero Gold, Platinum, and Platinum Eval only.)

WaveFormer Lite from SynaptiCAD allows graphical entering of HDL testbenches and manages multiple testbenches needed for different design configurations. The graphical testbench generation tool is ideal for designers unfamiliar with the process of creating testbenches, or for experts wanting to save time.

Actel Silicon Explorer accelerates device verification. Actel's antifuse FPGAs contain internal probe circuitry that provides built-in, no-cost, access to every node in a design, enabling 100% real-time observation and analysis of a device's internal logic without design iteration. The probe circuitry is accessed by Silicon Explorer, an easy to use integrated verification and logic analysis tool that attaches to a PC's standard COM port, turning the PC into a fully functional logic analyzer. Silicon Explorer is also available for download from <http://www.actel.com/custsup/updates/siliexp/index.html>.

Third-party software compatibility

Actel recommends that you use the integrated tools that come with Libero IDE. Consider the following issues when deciding to use other tools with Libero IDE.

Using Standalone ePD with Actel Libero IDE

Actel does not recommend that you use stand-alone versions of Innoveda's ePD or DxViewDraw with Libero. Do not install these and Libero's ViewDraw AE on the same PC. ViewDraw AE and ePD or DxViewDraw use similar environment variables and registry entries that will conflict. If you'd like to install ePD or DxViewDraw and Libero on the same machine, do not install ViewDraw AE during Libero installation.

A schematic created with ViewDraw AE should not be modified by a full version of ViewDraw. A full-block schematic created with ePD or DxViewDraw cannot be imported into ViewDraw AE.

Using ModelSim PE with Actel Libero IDE

If you try to use full standalone version of ModelSim tool with Libero, Libero's auto-generated library mapping may cause an error in the ModelSim environment. This is because the standalone ModelSim tool does not include compiled Actel libraries and does

not set up the compiled library tree in the same manner as the ModelSim AE tool.

Using other versions of Synplify with Actel Libero IDE

You can use a full version of Synplify with Libero. After purchasing a full version license of Synplify, set SYNPLICITY_LICENSE_FILE=<full version Synplify license location>\license.txt.

You can tell Libero IDE to look for a specific version of Synplify by creating a separate tool profile for each version. Choose which version to use in your project profile.

Using Precision RTL from Mentor Graphics with Libero IDE

You can use a full version of Precision RTL with Libero IDE. After purchasing Precision from Mentor Graphics, set up a tool profile for Precision RTL and add the tool to your project profile.

Using LeonardoSpectrum from Mentor Graphics with Libero IDE

You can use a full version of LeonardoSpectrum with Libero IDE. After purchasing LeonardoSpectrum from Mentor Graphics, set up a tool profile for LeonardoSpectrum and integrate the tool with your project profile.

Project Management

Libero IDE Design Flow

The Libero Design Flow consists of six steps:

Step One - Design Creation

Plan out your design and enter it as either HDL (VHDL or Verilog), structural schematic, or mixed-mode (schematic and RTL).

Step Two - Design Verification - Functional Simulation

After you have defined your design, you must verify that it functions the way you intended. After creating a testbench using WaveFormer Lite use the ModelSim VHDL or Verilog simulator to perform functional simulation on your schematic or HDL design.

Step Three - Synthesis/EDIF Generation

A design must be synthesized if the design was created using VHDL or Verilog. Use Synplify or Synplify Lite from Synplicity to generate your EDIF netlist. You can re-verify your design "post-synthesis" using the VHDL or Verilog ModelSim simulator used in step 2.

While all RTL code must be synthesized, pure schematic designs are automatically "netlisted" out via the Libero tools to create a structural VHDL or structural Verilog netlist.

Step Four - Design Implementation

After you have functionally verified that your design works, the next step is to implement the design using the Actel Designer software. The Designer software automatically places and routes the design and returns timing information. Use the tools that come with Designer to further optimize your design. Use Timer to perform static timing

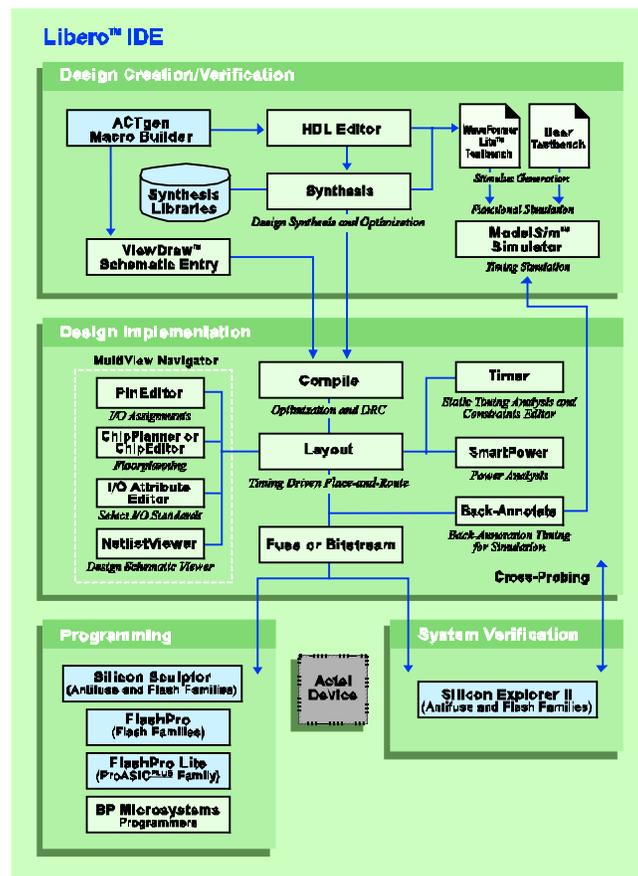
analysis on your design, ChipEditor or ChipPlanner to customize your I/O macro placement, PinEditor for I/O customization, SmartPower for power analysis, and NetlistViewer to view your netlist.

Five - Timing Simulation

After you are done with design Implementation, you can verify that your design meets timing specifications. After creating a testbench using WaveFormer Lite, use the ModelSim VHDL or Verilog simulator to perform timing simulation.

Step Six - Device Programming

Once you have completed your design, and you are satisfied with the timing simulation, create your programming file. Depending upon your device family, you need to generate a Fuse or Bitstream programming file.



Libero IDE Design Flow

Creating a new Libero project

Use the New Project Wizard to create a new Libero project.

To create a new project:

1. From the **File** menu, click **New Project**. The New Project Wizard starts.
2. Follow the instructions in the Wizard and click **Finish** when done.

Opening your Libero project

There are several ways you can open your Libero project files (.prj).

To open an existing project after opening Libero IDE:

1. From the **File** menu, click **Open Project**.
2. In **Look in**, navigate to the drive/folder where the .prj file is located and double-click to select it. (By default, Libero saves file to the Actelprj folder.)

Tip: The last five saved projects are available from the File menu. From the **File** menu, click **Recent Projects**, and then select the project to open.

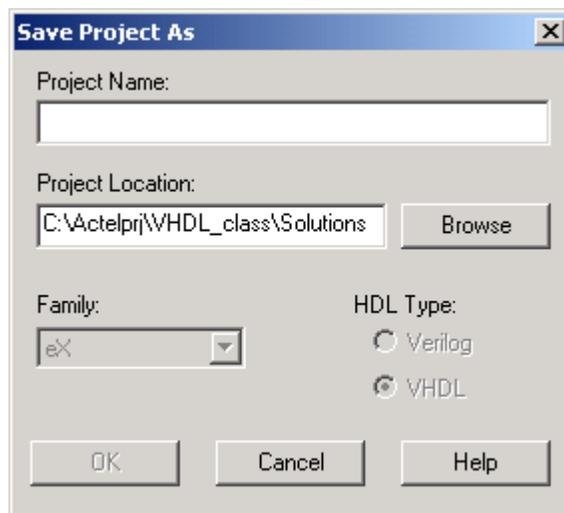
Tip: You can open an existing project by double-clicking the .prj file or dragging the .prj file over the Libero IDE desktop icon.

Saving a project with a new name

Your project is saved when you close the project. To save the project with another name, use the **Save Project As** command.

To save the project with a new name:

1. From the **File** menu, click **Save Project As**. The Save Project As dialog box opens.



Save Project As Dialog Box

2. Enter a new project name.
3. Enter a new project location, or click browse to specify a new location.
4. Click **OK**.

Closing and Exiting

Your project is automatically saved when closed. To save it with another name, use the Save Project As command.

To close a project:

1. From the **File** menu, click **Close Project**.

To exit Libero IDE:

1. From the **File** menu, click **Exit**.

Project sources (files)

Project sources are any design files that make up your design. These can include schematics, HDL files, simulation files, testbenches, etc. Anything that describes your design or is needed to program the device is a project source.

Source files appear in the Design Explorer window. The Design Hierarchy tab displays the structure of the design modules as they relate to each other, while the File Manager tab displays all the files that make up the project.

The design description for a project is contained within the following types of sources:

- Schematics
- HDL Files (VHDL or Verilog)

One source file in the project is the top-level source for the design. The top level-source defines the inputs and outputs that will be mapped into the devices, and references the logic descriptions contained in lower-level sources. The referencing of another source is called an *instantiation*. Lower level sources can also instantiate sources to build as many levels of logic as necessary to describe your design.

Some project sources can be imported.

Sources for your project can include:

Source	File Extension
Schematic	*.1-9
Verilog Module	.v
VHDL Entity	.vhd
ACTgen Macro	.gen
Testbench	.vhd
Stimulus	.tim
Programming Files	.afm, .prb

New Files

You can create new files from Libero. New file types include:

- Schematic
- ACTgen macro
- VHDL Entity
- VHDL Package file
- Stimulus
- Stimulus HDL file

To create a new file:

1. From the **File** menu, click **New**.
2. Select the File type and type a name.
3. Click **OK**. The appropriate application starts. The saved file is added to your Libero project.

Importing Files

Anything that describes your design, or is needed to program the device is a project source. These can include schematics, HDL files, simulation files, testbenches, etc. Import these source files directly into your Libero project.

To import a file:

1. From the **File** menu, click **Import Files**.
2. In **Files of type**, select the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.

Notes:

- Keep and import your VHDL package and behavioral and structural VHDL source files separately. Do not place your VHDL package into your source file.
- You cannot import a Verilog File into a VHDL project and vice versa.

File Types for Import

File Type	File Extension
ViewDraw Symbol	*.1-9
ViewDraw Schematic	*.1-9
Behavioral and Structural VHDL	.vhd, .vhdl
VHDL Package	.vhd, .vhdl
ACTgen Macro	.gen
Verilog Include	.h
Behavioral and Structural Verilog	.v
Stimulus	.vhd, .vhdl, .v
EDIF Netlist	.edn

Saving Files

Files and projects are saved when you close them.

To save an active file:

- From the **File** menu, click **Save**, **Save As**, or **Save All**.
- Click the **Save**  button in the toolbar.

Deleting Files

Files can be deleted from the current project or from the disk.

To delete a file from the project:

1. Select the **File Manager** tab in the **Design Explorer** window.
2. **Right-click** the file and select **Delete from Project**. The file remains on your disk.

To delete a file from your project and the disk:

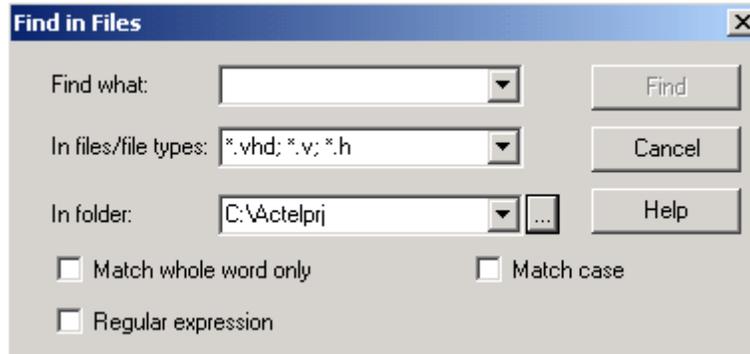
1. Select the **File Manager** tab in the **Design Explorer** window.
2. **Right-click** the file and select **Delete from Disk and Project**. The file is deleted from your disk and is no longer part of any project.

Finding files

Use the **Find In Files** dialog box to search for files.

To find a file:

1. From the **Edit** menu, click **Find**. The **Find In Files** dialog box appears.



Find In Files

2. Select the properties for your search.
 - **Find what:** Type a word string in the Find what text field.
 - **In files/file types:** Select a file type.
 - **In folder:** Select a folder.
 - **Match whole word only:** Select to match the whole word only.
 - **Regular expression:** Select to recognize Microsoft Word-style expressions.
 - **Match case:** Select to search for case-sensitive occurrences of a word or phrase. This limits the search so it only locates text that matches the upper- and lowercase characters you enter.
3. Click **Find**. The results appear in the Find In Files tab in the log window. Click the file name in the log window to open the file.

```

Searching for 'entity'...
C:\Actelprj\count32\simulation\presynth\cntadd\_primary.vhd(3): entity cntadd is
1 occurrence have been found.

```

Output | Errors | Warnings | Info | **Find In Files**

Found In Files Tab in Log Window

Libero project options

Use the Options dialog box to specify your project settings for the currently open project.

From the **Options** menu, click **Project Settings** to open the Options dialog box. View and edit the preferences and Click OK. To revert to the default settings, click Default.

Options include:

- Project Settings
- Simulation
- Programming

Libero project settings

Use the Project Settings tab in the Options window to change the device family.

To change the device family:

1. From the **Tools** menu, click **Options**. The Options dialog box appears .
2. Select a family from the list and click **OK**.

Note:

- You cannot change the device family if the project contains ACTgen files, structural files, or an .adb file.

Setting your simulation options

You can set a variety of simulation options for your project.

To set your simulation options:

1. From the **Options** menu, click **Project Settings**.
2. Click **Simulation**.
3. Select your options and click **OK**.
 - **Use automatic do file:** Select if you do not want Libero to initialize ModelSim.
 - **User defined Do file:** Enter the do file name or click the browse button.
 - **Compile VHDL Package files:** Select to compile VHDL package files using ModelSim AE.
 - **Include Do file:** Select to execute the wave.do or other specified Do file. Use the wave.do file to customize the ModelSim Waveform window display settings.
 - **Simulation Run Time:** Specify how long the simulation should run in ns. If the value is 0, or if the field is empty, there won't be a run command included in the run.do file.
 - **Testbench entity name:** Specify the name of your testbench entity name. Default is "testbench," the value used by WaveFormer Lite.
 - **Top Level instance name in the testbench:** Default is "<top_0>," the value used by WaveFormer Lite. Libero replaces <top> by the actual top level macro when ModelSim is run.
 - **Vsim Command Type:** Select Min, Typical (Typ), or Max
 - **Resolution:** The default is family specific, but you can customize it to fit your needs.

Family	Default Resolution
ACT1, ACT2, ACT3	1 ns
MX	1 ns
DX	1 ns
SX, SX-A	1 ns
eX	1 ns
Axcelerator	1 ps
ProASIC	1 ps
ProASIC <small>PLUS</small>	1 ps

- **Vsim additional options:** Text entered in this field is added to the vsim command.
- **Default:** Restores factory settings.

Programming

If you did not install FlashPro as part of the Libero installation process, you can use the Programming tab in the Options dialog box to integrate FlashPro with your project.

To integrate FlashPro with your project:

1. From the **Options** menu, click **Project Settings**.
2. Click the Programming tab.
3. Fill in the options:
 - **Location:** Specify the location of the FlashPro executable.
 - **Additional Parameters:** Additional parameters of the command line.
 - **Jar file location:** Specify the location of the FlashPro jar file.

4. Click OK.

Setting your project profile

Each Libero IDE project can have a different profile, enabling you to integrate different tools with different Libero projects.

To set or change your project profile:

1. From the **Options** menu, click **Profile**.
 - **To add a tool:** Click **Add** and select which type of tool (synthesis, stimulus, or simulation). Fill out the tool profile and click **OK**.
 - **To change a tool profile:** After selecting the tool, click to **Edit** to select another tool, change the tool name, or change the tool location.
 - **To remove a tool from the project:** After selecting a tool, click **Remove**.
 - **To restore the tool profiles shipped with Libero:** Click **Restore Defaults**.
2. When you are done, click **OK**.

VHDL Package Files Organization

It is important to preserve the VHDL package file sequence if your VHDL package files are interdependent.

To specify the VHDL package file order:

1. From the **Options** menu, click **VHDL Package File Organization**. The VHDL Package Files dialog box appears.
2. Use the up and down arrows to specify the order, or drag the files into order.
3. Select **Simulation** or **Synthesis** if you want the file included when simulation or synthesis is run.
4. Click **OK**.

Setting preferences

Use the Preferences dialog to customize Libero to your needs.

To set your preferences:

1. From the **File** menu, click **Preferences**.
2. Specify your preferences on each of the tabs.

Internet Tab

Proxy Tab

File Preferences Tab

Log Window

Text Editor

3. Click **OK**.

Note

- These preferences are stored on a per user basis. These preferences are not project specific.

Updates

Actel strongly recommends that you check at least once a week for fixes, updates, and enhancements for your Actel software.

The Updates tab in the Preferences dialog box allows you to set your automatic software update preferences.

To set your automatic software update preferences:

1. From the **File** menu, click **Preferences** and **Updates**.
2. Choose one of the following options.

- **Automatically check for updates at startup:** Select to be notified of updates when you start Designer.
- **Remind me to check for updates at startup:** Select to be asked if you want to check for a software update when you start Designer.
- **Do not check for updates or remind me at startup:** Select if you do not want to check for software updates at startup.

To manually check for software updates, from the **Help** menu, click **Check for Software Updates**.

3. Click **OK**.

Note:

- This feature requires an internet connection.

Setting Your Proxy

An FTP connection is used to update some data files. Use the Proxy tab in the Preferences dialog box to enter your proxy name if you use a proxy server.

1. From the **File** menu, click **Preferences**.
2. Click the **Proxy** tab.
3. If you use a Proxy server, click the check-box and enter the name.
4. Click **OK** to dismiss the Preferences dialog box.

Libero file association

Several programs, including Libero, create files with the .prj extension. If you want Libero to start whenever you double-click on a .prj file, you need to set up Libero as the default editor for .prj files.

To make Libero the default editor for .prj files:

1. From the File menu, click **Preferences**.
2. Click the **File Association** tab.
3. Check the box to associate Libero with .prj files.

Setting your log window preferences

Errors, Warnings, and Informational messages are color coded in the log window. You can change the default colors by using the log Window tab in the Preferences dialog box.

To change colors in the log window:

1. From the **File** menu, choose **Preferences**.
2. Click the **Log Window** tab in the Preferences Dialog Box.
3. Select your new default colors and click **OK**.

The default color settings for the log window are:

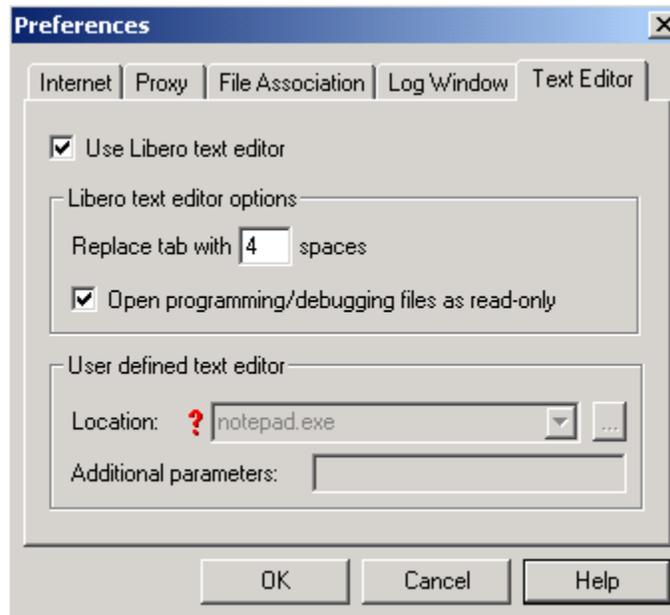
Message Type	Colors
Errors	Red
Warnings	Light Blue
Informational	Black
Linked	Dark Blue

Text editor

You can use the Libero HDL text editor or another text editor.

To set your text editor preferences:

1. From the **File** menu, click **Preferences**.
2. Click **Text Editor**.



Preferences: Text Editor

2. Set your options and click **OK**.

Libero text editor options

- **Use Libero text editor:** Select to use the Libero HDL text editor.
- **Replace tab with spaces:** Enter the number of spaces you want entered when using the tab key.
- **Open programming/debugging files as read-only:** Select to specify read-only permission to .stp and .prb files.

- **User defined text editor:** Unselect use Libero text editor to activate this area. Enter the .exe location of the text editor.
- **Additional parameters:** Use to specify other settings to pass to the text editor. Typically, it is not necessary to modify this field.

Libero's Project Manager

Libero's Project Manager workspace integrates the needed design tools, streamlines the design flow, manages all design and log files, and passes necessary design data between tools.

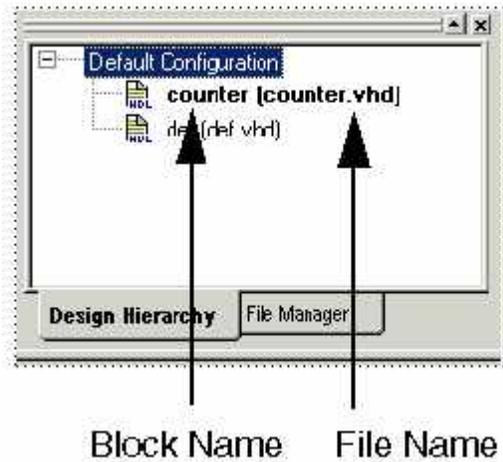
The Design Explorer Window, located in the upper left, consists of the Design Hierarchy and File Manager windows. Below is the Process Window. The HDL Editor fills up the right side of the Project Manager, while the Log Window is found at the bottom.

Libero also includes toolbars and menus.

Design Hierarchy

The Design Hierarchy tab displays a hierarchical representation of the design based on the source files in the project. Libero IDE continuously analyzes and updates source files and updates the hierarchy. The Design Hierarchy tab displays the structure of the design modules as they relate to each other.

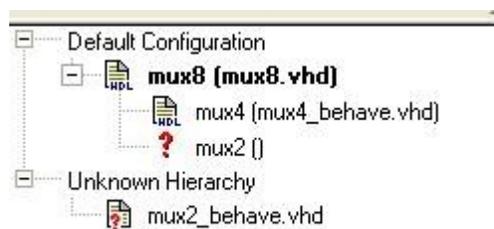
The corresponding file name (the file that defines the block) next to block name in parentheses.



Design Hierarchy - Block and File Names

More information about the block can be found by right-clicking it and selecting **Properties**. The Block Properties dialog box displays block properties including, file path, created date, and last modified date.

Files that cannot be read by Libero are identified with red question marks.



Design Hierarchy - Unknown Hierarchy

All integrated source editors are linked with Libero's Project Manager. If a source is modified and the modification changes the hierarchy of the design, the Design Hierarchy automatically updates to reflect the change.

If you want to update the design hierarchy, choose **Refresh** from the **Edit** menu.

To open a source:

Double-click a source in the Design Hierarchy to open it. Depending on the block type and design state, several possible options are available from the right-click menus.

File Manager

The folders in the File Manager are like the ones in the Windows Explorer. These folders reside on your computer and support relative paths - paths from the project folder to files in your folders. This folder structure makes it easier to organize your files.

The File Manager window displays all the files associated with your project. Files are grouped by type.

- block symbol files
- schematic files
- HDL files
- ACTgen Macros
- stimulus files
- design implementation files

Right-clicking a file in the File Manager provides a menu of available options specific to the file type. You can also delete files from the project by selecting **Delete from Project**

from the right-click menu.

Tip: You can drag files in the File Manager to re-order them.

Process Window

The process window displays all available tools involved in the design process.

This window shows you the current state of your design by activating and highlighting tools at appropriate times in the design process, while graying out tools that are not yet available. Green checks indicated successfully completed steps.

Double-click a tool to start it. Right-click a tool to access the right-click menu, which provides all the available processes you can start with the tool.

HDL Editor Window

The HDL Editor targets the creation of HDL code. It supports VHDL and Verilog with color, highlighting keywords for both HDL languages.

Note

- To avoid conflicts between changes made in your HDL files, Actel recommends that you use one editor for all of your HDL edits.

Log Window

Colors and Symbols

For ProASIC and ProASIC^{PLUS} families the log window displays notes and warnings. For Antifuse families, the log window displays error, warning, and informational messages. Messages are represented by symbols and color coded. The default colors are:

Type	Color
------	-------

Error	Red
Warning	Blue
Information	Black

The colors can be changed by using the Preferences dialog box.

Output, Error, Warning, and Info Tabs

The Output tab displays all messages. Use the errors, warnings, or informational tabs to filter for just those messages. The views within the error, warnings, and info displays are reset when a new command is executed or a new design is opened. To see a complete history of your design session, click the output tab.

Linked Messages

Error and warning messages that are dark blue and underlined are linked to online help to provide you with more details or helpful workarounds. Click them to open online help.

HDL Entry

Using the HDL Editor

The HDL Editor is a text editor designed for editing HDL source files. In addition to regular editing features, the editor provides a syntax checker.

You can have multiple files open at one time in the HDL Editor workspace. Click the tabs to move between files.

The easiest way to use the HDL editor is to maximize its space by closing the Design Explorer and Process windows. Close these windows by selecting them in the View menu.

Editing

Editing functions are available in the Edit menu. Available functions include cut, copy, paste, find and replace. These features are also available in the toolbar.

Saving

You must save your file to add it to your Libero project. Click **Save** in the **File** menu, or click the Save icon in the toolbar.

Printing

Print and Print Preview functions are available from the File menu and the toolbar.

Note

- To avoid conflicts between changes made in your HDL files, Actel recommends that you use one editor for all of your HDL edits.

Creating new HDL files

To create an HDL file:

1. Open your project.
2. From the **File** menu, click **New**.
3. Click **VHDL** and type a file name in the Name field. Click **OK**. (Do not enter a file extension; Libero adds one for you.) The HDL Editor workspace opens.
4. After creating your HDL file, save your file to the project by clicking **Save** from the **File** menu. Your HDL file is saved to your project, appearing in the File Manager.

Opening an HDL source file

To open an HDL source file:

1. From the **File** menu, click **New**.
2. From **Files of Type**, select **HDL File (*.vhd, *.vhdl)**.
3. In **Look in**, navigate to the drive/folder where the .hdl file is located.
4. Select your file and click **Open**. Libero opens your file in the HDL Editor

Importing HDL source files

Import your HDL file into your project just as you would any source file.

To import an HDL source file:

1. From the **File** menu, click **Import Files**.
2. In **Files of type**, select the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.

HDL Syntax Checker

After you are done creating your HDL file, use the HDL Syntax Checker to help validate an HDL file after editing the HDL code.

To run the syntax checker:

1. In the **Libero File Manager**, right-click an HDL file and click **Check HDL File**.
2. The syntax checker parses the selected HDL file and looks for typographical mistakes and syntactical errors. Warning and error messages for the HDL file appear in the Libero Log Window.

Commenting Text

You can comment text as you type in the HDL Editor, or you can comment out blocks of text by selecting a group of text and applying the Comment command.

To comment or uncomment out text:

1. Type your text.
2. Select the text.
3. From the **Edit** menu or right-click menu, click **Comment Out** or **Uncomment**.

Using ACTgen macros

Use ACTgen to:

- create high level modules, such as counters, multiplexers, multipliers, etc. that are optimized for Actel FPGAs.
- create system level building blocks, such as filters, FIFOs and memories.

These can be instantiated into your schematic, Verilog design, or HDL design.

To use ACTgen with your HDL design:

1. **Add the ACTgen macro to your Libero project.**
 - From the Libero **File** menu, click **New**.
 - In the New File dialog box, select ACTgen macro, type a name, and click **OK**. ACTgen starts.
 - Select your macro type from the left Macro list box. The appropriate options appear. Select a tab and fill in the fields. Click **Generate**.
 - In the Save As dialog box, leave the default selections and click **Save**. The file is added to your Libero project, appearing in the Design Hierarchy.

2. **Instantiate the module in your HDL design.**

Schematic Entry

ViewDraw AE

ViewDraw AE is a special version of ViewDraw. Use it for schematic entry with Libero IDE.

Actel does not recommend that you use stand-alone versions of ePD or DxViewDraw from Mentor Graphics with Libero IDE. Do not install these and ViewDraw AE on the same PC. ViewDraw AE and ePD or DxViewDraw use similar environment variables and registry entries that will conflict. If you'd like to install ePD or DxViewDraw and Libero on the same machine, do not install ViewDraw AE during Libero installation.

A schematic created with ViewDraw AE should not be modified by a full version of ViewDraw. A full-block schematic created with ePD or DxViewDraw cannot be imported into ViewDraw AE.

Note: The full online help system for ViewDraw AE can be accessed by opening ViewDraw AE and clicking the **Help** menu.

Schematic guidelines

When creating your schematics, follow these guidelines.

Using Hierarchical Connectors

Hierarchical connectors are required in any schematic design for external ports for the top level. Hierarchical connectors must be used also on ports of sub-modules.

I/O Pads

I/O pads can be automatically inserted by running synthesis. For schematic designs that only use Actel primitives (no HDL blocks), you can manually insert ALL I/O pads if you do not want to go through the synthesis flow.

Naming Conventions

1. Project names should be less than 8 characters.
“my_top” is acceptable
“my_top_level” is not
2. Do not use spaces in:
 - Project names
 - Instance names
 - Net names
 - Port names
3. Do not use any special characters in any of your naming, such as: ~ ! @ # \$ % ^ & * () = + { } | \ / < > ? ` ' " " , . or spaces.
4. The "inverted" net property is not supported.
5. If you want to rip out scalar bits from a bus, use [] for scalar bit naming. For example, Bus[15], Bus[14], ..., Bus[0].
6. Do not use numbers at the beginning or the end of any names. ViewDraw AE regards Bus[1] as equivalent to Bus1.

Scalar bit Bus1[15] of Bus1[15:0] conflicts with scalar bit Bus11[5] of Bus11[15:0] during netlist generation.

If you want to use numbers to distinguish related nets, numbers can be used followed by letters at the end, for example: NET1N, or Bus1A[15:0].

7. Multi-dimensional busses are not supported. For example, do not use naming "Bus[0:3][0:3]" in ViewDraw AE.

Creating a schematic source file

Use ViewDraw AE to create your schematic source files.

To create a schematic source file:

1. From the **File** menu, click **New**.
2. Select **Schematic** and type a name for your schematic file in the Name field. Click **OK**. ViewDraw AE starts.
3. Using ViewDraw AE, create your schematic.
4. When you are done, click **Save+Check**. The Save+Check command creates your WIR file. When Save and Check is complete, the message “Check complete, 0 errors and 0 warnings in project <name>” appears in the status bar. You must select Save & Check. Only selecting Save will not generate the needed WIR file for that block.
5. (Optional) Run connectivity checker. **Right-click the schematic file** in the File Manager tab and click **Check Schematic**. The connectivity checker checks the connectivity of the wir file. Errors and warnings appear in the log window.
6. From the File menu, click **Exit**. The schematic is saved to your project in Libero, appearing in both the File Manager and the Design Hierarchy tabs.

Importing schematics

You can import any schematic created with ViewDraw AE.

To import a schematic file:

1. From the **File** menu, click **Import Files**.
2. In **Files of type**, select Schematics.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**. The schematic is imported into your

project and appears in the File Manager, under Schematic files.

To open the schematic, double click on ViewDraw AE in the Process window, or right-click the file in the File Manager and select, **Open Schematic**.

Notes:

A schematic created with ViewDraw AE should not be modified by a full version of ViewDraw. A full-block schematic created with ePD or DxViewDraw cannot be imported into ViewDraw AE.

Opening a schematic source file

Use ViewDraw AE to edit your schematic files.

To open your schematic file:

1. **Open** your project in Libero IDE.
2. Double-click the schematic file in the File Manager or Design Hierarchy windows. ViewDraw AE opens with the file loaded.
3. From the **File** menu, click **Save+Check** to create the required files for netlist generation. When Save + Check is complete, the Status Bar will say "Check complete, 0 errors and 0 warnings in project <name>." You must select Save +Check. Only selecting Save will not generate the needed WIR file for that block.
4. From the **File** menu, click **Exit**. The schematic is saved to the project, appearing in both the File Manager and Design Hierarchy tabs. Your schematic file is updated in Libero.

Using ACTgen macros

Use ACTgen to:

- create high level modules, such as counters, multiplexors, multipliers, etc. that are optimized for Actel FPGAs.
- create system level building blocks, such as filters, FIFOs and memories.

These can be instantiated into your schematic, Verilog design, or HDL design.

To generate macros for your schematic:

1. **Add the ACTgen macro to your Libero project.**
 - From the Libero **File** menu, click **New**.
 - In the New File dialog box, select ACTgen macro, type a name, and click **OK**. ACTgen starts.
 - Select your macro type from the left Macro list box. The appropriate options appear. Select a tab and fill in the fields. Click **Generate** to create an HDL representation of the macro.
 - In the Save As dialog box, leave the default selections and click **Save**. The file is added to your Libero project, appearing in the Design Hierarchy.
2. **Create the Symbol.** In the Design Hierarchy, right-click the ACTgen module and choose **Create Symbol**. The symbol is created, appearing in the File Manager, under Block Symbol files.
3. **Use the Symbol.**
 - Start ViewDraw.
 - From the **Add** menu, click **Component**.
 - Select the new symbol, then drag and drop it onto your schematic.

Design Constraints

About Design Constraints

The Designer software supports both physical and timing constraints. Constraints can be set by either using Actel's interactive tools or by importing constraint files directly into Designer.

Physical Constraints

Designer supports two types of physical constraints: I/O Assignments and Location and Region Assignments.

I/O Assignments

Use PinEditor to manually place and configure your I/Os. Or, assign I/O locations automatically by importing one of the following constraint files into Designer:

- GCF (Flash families)
- PDC (Axcelerator family)
- PIN file (SX-A, eX, SX, MX, 3200DX, 1200XL, ACT3, ACT2, ACT1 families)

Location and Region Assignments

Use ChipEditor to view and manually change location assignments. Use ChipView to view the placement and routing in Flash designs.

You can also assign location constraints and enter region constraints by importing one of the following constraint files into Designer:

- GCF (Flash families)
- PDC (Axcelerator family)

PinEditor

The PinEditor tool provides a graphical application for displaying and configuring I/O assignments and attributes.

Use PinEditor to:

- assign I/O macros to pins
- fix pin assignments that have automatically been assigned during layout
- view and print pin assignments
- assign I/O standards to banks, for families that utilize I/O banks
- edit I/O attributes, such as I/O standards, slew, and capacitance
- assign VREF pins, for I/O standards that require an input reference voltage

To start PinEditor:

There are three ways to start PinEditor:

- Click **PinEditor** in the Designer main flow window
- From the **Tool** menu, click **PinEditor**
- Click the **PinEditor** toolbar button in Designer

PinEditor Interface

Package window

PinEditor's Package window displays pins, I/O macro assignments, and I/O Banks (Axcelerator family only).

The Package window is integrated with PinEditor windows and list boxes. If you select an assigned pin in the Package window, the pin location is highlighted in the World

View window and the I/O macro name is selected in the Assigned list box and the I/O Attribute Editor.

The Package Window displays detailed information about each pin, including:

- pin number
- special pin properties, such as JTAG, clock, ground, or power
- assigned I/O macro name, if any
- pin type, represented by color

Color Manager

Use the Color Manager to customize the colors used to display the package in ChipEditor and PinEditor. The Color Manager specifies the display colors for I/O banks, I/O FIFO Blocks, RAM tiles, Core tiles, clusters, super clusters, and nets.

To customize the colors in the Package window:

1. From the **View** menu, click **Color Manager**.
2. In the Color Manager dialog box, click the color box in front of the item you wish to customize, or click the I/O bank you wish to change. The color pallet is displayed.
3. Select a color and click **OK**. The new color will appear in the Color Manager dialog box.
4. After you are done customizing your colors using the Color Manger dialog box, click **OK**.

World View window

The World View window's default location is under the Assigned list box. Use the World View window to control which portion of the package is displayed in the Package window. The blue rectangle (known as the Package rectangle) represents the package. The green rectangle (known as the Viewing rectangle) represents the currently displayed area in the Package window.

To display another part of the package, use the left mouse button to drag the Viewing rectangle to the area on the Package rectangle you would like to display. To specify a new display area, use the right mouse button to stroke out a new Viewing rectangle on the

Package rectangle.

I/O Attribute Editor

The default location of the I/O Attribute Editor is below the Package and World View windows. The I/O Attribute Editor lists all assigned and unassigned I/O macros and their attributes in a spreadsheet format. Use the I/O Attribute Editor to view, sort, select, and edit these I/O attributes. Double-click a column heading to sort by that attribute. If you select a macro in the list boxes, the I/O attribute editor scrolls to highlight the selected macro.

Configure PinEditor List Boxes

Use the Configure List Box dialog box to customize what is displayed in the Assigned and Unassigned list boxes.

To configure the listboxes:

1. From the **View** menu, click **Configure List Boxes**.
 - **Filter Assigned and Unassigned Lists:** Entering a specific pin name in this field filters out all other pins in the Assigned and Unassigned List Boxes. Use the * wildcard to filter for groups.
 - **Show fixed and unfixed pins:** Selecting this causes all fixed and unfixed pins to be displayed in the Assigned list box.
 - **Show only fixed pins:** Selecting this filters out all unfixed pins from the Assigned list box.
 - **Show only unfixed pins:** Selecting this filters out all fixed pins from the Assigned list box.
2. Click **Apply** to see changes. When satisfied, click **OK**.

Making Pin Assignments

Use PinEditor to make and edit I/O macro pin assignments. Edits made in PinEditor are permanent, as long as they are fixed and committed.

Depending upon the design family, PinEditor opens as a stand alone tool or in the MultiView Navigator.

To assign an I/O macro to a pin using PinEditor in the MultiView Navigator interface:

1. Select the instance in the Logical or Physical tabs.
2. Drag the instance to the pin location.

If the location is a valid one, the macro is assigned and automatically fixed.

To assign an I/O macro to a pin using stand alone PinEditor:

1. Select the macro name in the Unassigned list box. The macro is simultaneously selected in the I/O Attribute Editor.
2. Assign the selected macro to a pin location using any one of these methods:
 - Drag the selected macro name from the Unassigned list box to the pin location in the Package Window. Valid pin locations are highlighted in the Package Window.
 - In the Edit menu, choose **Assign** to invoke the Assign mode. Then, select the pin location in the Package Window.
 - Click the **Assign** toolbar button to invoke the Assign mode and then select the pin location in the Package window.

- If you know the specific pin location, enter the pin assignment in the Pin# cell or select a valid placement from the drop-down menu.

If the location is a valid one, the macro is assigned and automatically fixed. The status bar displays information about invalid assignments. Choose *Extended Error Messages* from the Help menu for more information about specific error messages.

Note:

- If you assign a macro to a pin that has already been assigned a macro, the previously assigned macro becomes unassigned, even if its placement has been fixed.

Locking Pin Assignments

Designer does not alter locked pins during Layout. Designer recognizes pins as locked if they are:

- assigned manually using PinEditor
- assigned in a design schematic
- assigned using a pin file (non-Axcelerator ProASIC families)
- assigned using a PDC file (Axcelerator family only)

Locked pins are permanent, as long as you commit your locked pins to your design before you exit PinEditor.

To lock pins:

1. Select the pin(s) to lock in the Assigned list box, Package window, or I/O Attribute Editor. To select multiple pins, hold the ctrl key and select multiple pins with your mouse. To select all pins, choose **Select All** from the **Edit** menu.
2. In the **Edit** menu, click **Lock**. Or, using the I/O Attribute Editor, select the Locked check box.

Note:

- If you are using the I/O Attribute Editor, you can only fix one pin at a time.

To unlock a pin:

1. Select the pin(s) to unlock in the Assigned list box, Package window, or I/O Attribute Editor. To select multiple pins, hold the ctrl key and select multiple pins with your mouse. To select all pins, choose **Select All** from the **Edit** menu.
2. In the **Edit** menu, click **Unlock**. Or, in the I/O Attribute editor, uncheck the Locked check box.

Closing and committing pin assignments

Edits made in PinEditor are only temporary. If you wish to keep your pin assignments and I/O attribute changes, you must commit your changes before closing PinEditor.

To commit your pin assignments at any time, from the **File** menu, click **Commit**.

To commit your pin assignments when closing PinEditor click Yes when asked if you would like to commit changes made in PinEditor.

Committing your changes saves them to the "working" design for this Designer session only.

To permanently save changes made in PinEditor to your design file, (.adb) you must save your design. From the Designer **File** menu, click **Save**.

Using the I/O Attribute Editor

The I/O Attribute Editor displays I/O attributes in a spreadsheet format. Use the I/O Attribute Editor to view, sort, select, and edit standard and device-specific I/O attributes.

Standard attributes

The I/O Attribute Editor shows four standard attributes for all I/O macros:

- **Port Name** indicates the I/O macro name.
- **Macro Cell** indicates the type of I/O macro.
- **Pin #** indicates the current pin assignment.
- **Fixed**, if checked, indicates that you cannot change the current pin assignment during layout.

Axcelerator I/O attributes

Besides the standard attributes, the I/O Attribute Editor displays Axcelerator specific attributes. The list below includes a description of Axcelerator specific attributes.

1. **I/O Standard** indicates the I/O standard. Possible I/O standards include LVTTL, LVCMOS 2.5V, LVCMOS 1.8V, LVCMOS 1.5V, 3.3V PCI, LVDS, LVPECL, GTL+, HSTL Class I, SSTL3 Class I and II, and SSTL2 Class I and II. Information on these standards can be found in "Glossary" on page 51. Refer to the appropriate data sheet for information about I/O standards for different families.
2. **Slew** indicates the slew rate for output buffers. Generally, available slew rates are high and low. The output buffer has a programmable slew rate for both high to low and low to high transitions. The slow slew rate is incompatible with 3.3V PCI requirements. For the Axcelerator family, slew can only be edited for the LVTTL I/O standard.
3. **Resistor Pull** indicates the resistor pull: NONE, weak pull-up, Weak pull-down. The default value is NONE. The only exception to this is an I/O that exists in the netlist as a port, is not connected to the core, and is configured as an Output Buffer. In that case, the default setting will be for a weak pull-down.
4. **Hot Swap** indicates if the pin is hot swappable. The device, the
5. **I/O standard** specified, and the selected voltage determine this read-only attribute. All the I/O standards except 3.3V PCI are hot-swap compatible and 3.3V tolerant.
6. **Loading** (pf) indicates the output-capacitance value based on the I/O standard selected in the I/O Standard cell.
7. **Input Delay** is set to "on" by checking the box.
8. **Output Drive Strength** can be set to 8, 12, 16, 24 in mA, weakest to strongest. The LVTTL output buffer has four programmable settings of its drive strength. Other I/O standards have full strength.
9. **Bank Name** displays the bank name. This cannot be edited.

SX-A and RTSX-S I/O Attributes

Besides the standard I/O attributes, the I/O Attribute Editor displays device-specific attributes. Device-specific attributes vary by device and only supported attributes are displayed. The list below includes a description of SX-A and RTSX-S specific attributes.

1. **I/O Standard** indicates the I/O standard. Possible I/O standards include LVTTL/TTL, PCI, CMOS, Custom. Information on these standards can be found in "Glossary" on page 51. Refer to the appropriate data sheet for information about I/O standards for different families.
2. **IO Threshold** indicates compatible threshold level for inputs and outputs, either CMOS, TTL, or PCI.
3. **Slew** indicates the slew rate for output buffers. Generally, available slew rates are high and low. The output buffer has a programmable slew rate for both high to low and low to high transitions. The slow slew rate is incompatible with 3.3V PCI requirements.
4. **Resistor Pull** indicates the resistor pull at power-up time: NONE, weak pull-up, Weak pull-down. This state is of short duration and does not stay. The default value is NONE. The only exception to this is an I/O that exists in the netlist as a port, is not connected to the core, and is configured as an Output Buffer. In that case, the default setting will be for a weak pull-down.
5. **Hot Swap** indicates if the pin is hot swappable. The device, the I/O standard specified, and the selected voltage determine this read-only attribute. All the I/O standards except 3.3V PCI are hot swap compatible and 5V tolerant.
6. **Loading** (pf) indicates the output-capacitance value based on the I/O standard selected in the I/O Standard cell. If you have selected custom in the I/O Standard field, you can modify the capacitance value to any integer value that accurately reflects the capacitive loading on the Actel device pins.

ProASIC and ProASIC ^{PLUS} I/O Attributes

Besides standard I/O attributes. the I/O Attribute Editor also displays device-specific attributes. Device-specific attributes vary by device and only supported attributes are displayed. The list below includes a description of ProASIC ProASIC ^{PLUS} and specific

attributes.

1. Loading (pf) indicates the output-capacitance value based on the I/O standard in the I/O Standard cell. The loading selected is applied to all outputs.

About I/O Banks

For devices that support multiple I/O standards, I/Os are grouped onto I/O banks around the chip.

The Axcelerator Family has 8 I/O banks that surround the chip, two per-side, numbering 0-7. The I/O banks are color coded for quick identification. (Colors can be changed using the Color Manager.)

Each I/O bank has a common:

- VCCI, the supply voltage for its I/Os
- VREF, the reference voltage bus (for voltage-referenced I/O standards)

Only one VREF value can be assigned to each I/O bank. Only I/Os compatible with both the same VCCI and VREF standards can be assigned to the same bank.

Assigning technologies to I/O Banks

To assign technologies to banks:

1. Select an I/O bank.
2. From the **Edit** menu, click **I/O Bank Properties**.
3. In the Configure I/O Dialog Box, select your options and click **Apply**. The I/O bank is assigned the selected standards. Any I/O of the selected types can now be assigned to that I/O bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are removed.
4. If VREF pins can be assigned, the Assign VREF Pins button will highlight.
5. Assign I/O standards to other banks by selecting the banks from the list and assigning standards. Any banks not assigned I/O standards use the default standard selected in the Device Selection Wizard.
6. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

I/O Bank options

When assigning technologies to your I/O banks, use the Configure I/O Bank dialog box.

Options include:

Select Technologies

Selecting a standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Further selecting GTL (3.3V) disables SSTL3 as an option because the VREFs of the two are not the same.

Assign VREF Pins

After you have selected your technology, click Apply. If VREF pins are required, this button becomes activated. Click to assign VREF pins. You must assign VREF pins at least once.

Click More Attributes to set the following:

Low Power Mode (Optional)

Select Enable Input Buffers or Enable Output Buffers. These are not required. This feature is not supported in the RTAX-S family.

Input Delay

Drag the slider bar to your desired delay. The delay is bank specific. Drag the meter to your desired delay index. The delay code and typical value appear. Click **View All Delays** to see all the delay values (Best, Worst, Typical, Rise-Rise, Fall-Fall) for the input delay selected. A technology must be selected in order to see the input delays. Click **OK** to dismiss the View All Delays dialog box. This feature is not supported in the RTAX-S family.

Specifying I/O bank voltage

You can directly specify voltages for each I/O bank by doing one of the following:

- Using the Assign Technologies to I/O Banks dialog box
- Placing an I/O of a particular technology in an I/O bank that has not been assigned a voltage
- Using the command `set_iobank` in a PDC File

Location and Region Assignments

Location and region assignments can be made using ChipEditor and ChipPlanner.

Using ChipEditor

ChipEditor is a graphical application for viewing and placing I/O and logic macros. This tool is particularly useful when you need maximum control over your design placement.

ChipEditor supports the following families: ACT1, ACT2, ACT3, DX, MX, SX, SX-A, and eX families. For all other families, use ChipPlanner.

Use ChipEditor to:

- View macro placements made during layout
- Place, unplace, or move macros
- Fix I/O macro placements
- View net connections using a ratsnest, minimum spanning tree, or route view
- View architectural boundaries
- View and edit silicon features, such as I/O banks
- Cross probe with Silicon Explorer to select probes
- View placement and routing of paths when used with Timer

Using ChipPlanner

ChipPlanner is a graphical application for viewing and placing I/O and logic macros. You can also use it for floorplanning. This tool is particularly useful when you need maximum control over your design placement.

Note: ChipPlanner does not support ACT1, ACT2, ACT3, MX, DX, eX, SX, SX-A. Use ChipEditor instead.

Use ChipPlanner to:

- View macro placements made during layout
- Place, unplace, or move macros
- Fix I/O macro placements
- View net connections using a ratsnest or route view
- View architectural boundaries
- View and edit silicon features, such as I/O banks
- Cross probe with Silicon Explorer to select probes
- View placement and routing of paths when used with Timer
- Create and assign macros or nets to regions

Starting ChipPlanner

ChipPlanner requires a compiled design. Therefore, you can only start ChipPlanner if your design has been compiled. If you start ChipPlanner before compiling your design, Designer guides you through the compile process before opening ChipPlanner.

You must start ChipPlanner from Designer.

To start ChipPlanner do one of the following:

- Click **ChipPlanner** in the Designer design flow window
- Click the **ChipPlanner** toolbar button in Designer

ChipPlanner opens in the MultiView Navigator interface.

The MultiView Navigator opens with ChipPlanner active.

Starting and Exiting ChipEditor and ChipPlanner

To start ChipEditor or ChipPlanner:

1. If you have not done so, Compile your design.
2. From the **Tools** menu, click **ChipEditor** or **ChipPlanner**. ChipEditor starts in a separate window. If you are running ChipPlanner, the MultiView Navigator starts with ChipPlanner active.

Tip: You can also start these tools by clicking ChipEditor or ChipPlanner in the Designer design flow window

Exiting ChipPlanner:

1. From the **File** menu, click **Close**.
2. To close the MultiView Navigator, from the **File** menu, click **Exit**.

Exiting ChipEditor:

1. **From the File menu, click Exit. If you haven't committed your changes, you will be asked if you want to commit your changes.**

Committing

Changes made in **ChipEditor** or **ChipPlanner** are not permanent until you use the Commit command. The Commit command saves your changes to your design session. Changes are not reversible. To permanently save your changes, you must save your design in Designer.

To commit your changes:

1. From the **File** menu, click **Commit**.

Logic assignment

Manually assigning logic is an optional methodology to help you improve the performance and density of your design. If your design requires refining or customizing, ChipPlanner and ChipEditor provide maximum control to achieve optimum results.

To manually assign logic, use ChipPlanner and ChipEditor before and after Layout.

Assigning and unassigning logic

You do not need to manually assign logic in your design. However, should you have specific design requirements, ChipEditor and ChipPlanner allow you to have maximum control over your design.

To assign logic using ChipPlanner:

1. Select the logic in the Physical tab.
1. Drag the logic to the desired location. As you drag, valid assignment locations are highlighted. To remove the assignment, from the **Edit** menu, click **Undo**.

If the logic assignment is valid, the logic is assigned and locked. To save changes for this design session, commit your changes when exiting the MultiView Navigator.

Note: Assigning logic to a location that already has logic unassigns the previously assigned logic, even if its assignment was locked.

To assign logic using ChipEditor:

1. Select the logic in the Unassigned list box.
2. Drag the logic to the desired location in the ChipEditor Window.

If the logic placement is valid, the logic is placed. To remove the placement, from the **Edit** menu, click **Undo**.

Error messages in the status bar notify you about invalid placement attempts. Choose Extended Error Messages from the Help menu for more details on a specific error message. If you want to ensure that the logic is not moved during layout, you must Lock the logic assignment and commit your changes when exiting ChipEditor.

Note: Assigning logic to a location that already has logic unassigns the previously assigned logic, even if its assignment was locked.

To assign multiple logic macros:

1. While holding down the CTRL or SHIFT key, select the logic in the order you want it placed.
2. From the **Edit** menu, click **Assign**.
3. One by one, select the desired location. The macros are placed in the order selected.

To unassign logic:

1. Select the logic.
2. From the **Edit** menu, click **Unassign**.

To unassign multiple logic macros:

1. Hold down the CTRL or SHIFT key and select the logic you want to unplace. To select all logic, choose **Select All** from the **Edit** menu.
2. From the **Edit** menu, click **Unassign**.

Moving logic

You can move logic that was placed manually or automatically during Layout.

To move logic:

1. Select the logic.
2. Drag the logic to the new location.

Tip: To remove the placement, from the **Edit** menu, click **Undo**.

Locking logic

Locked logic is not moved during Layout. Locked logic only becomes permanent if you commit the changes to your design before exiting ChipEditor or ChipPlanner.

To lock macros:

1. Select the macro to lock. To select multiple macros, hold the CTRL key and select multiple macros with your mouse. To select all macros, choose **Select All** from the **Edit** menu.
2. From the **Edit** menu, click **Lock**.
3. From the **File** menu, click **Commit** to make the changes permanent and update your .adb file.

To unlock a macro:

1. Select the macro. To select multiple macros, hold the CTRL key and select multiple macros with your mouse. To select all macros, from the **Edit** menu, click **Select All**.
2. From the **Edit** menu, click **Unlock**.

About I/O FIFOs

In the Accelerator family, every I/O can have a dedicated PerPin FIFO with a fixed depth of 64 bits. Each PerPin FIFO can be individually controlled; in which case the flag logic has to be built into the FPGA. Alternatively, you can use the embedded I/O FIFO Controllers to control sets of I/Os.

The embedded I/O FIFO Controllers can control sets of I/Os. These sets are known as I/O FIFO Blocks.

Note: RTAX-S does not support PerPin FIFOs and I/O FIFO Controllers.

Assigning I/O FIFO Controllers

Manual assignment of the embedded I/O FIFO Controllers before running Layout is required.

I/O FIFO Controller Considerations:

- All I/Os with a dedicated FIFO that is controlled by the same I/O FIFO Controller must be assigned in the same I/O FIFO Block as the I/O FIFO Controller.
- All I/Os with a dedicated FIFO that is individually controlled can be assigned anywhere.
- I/Os with a dedicated FIFO cannot be controlled by different embedded I/O FIFO Controllers in the same I/O FIFO Block.
- The I/O FIFO Controllers must be assigned before running Layout. They cannot be automatically assigned.

The above does not change the banking rules (i.e. any I/Os assigned to a bank must have compatible technologies). While the I/O FIFO or I/O FIFO Controller does not have a technology requirement, the I/O that contains the dedicated FIFO does.

Note: RTAX-S does not support PerPin FIFOs and I/O FIFO Controllers.

To assign an I/O FIFO Controller:

1. Select the I/O FIFO Controller you want to assign.
2. Drag the I/O FIFO Controller to the desired location.

Floorplanning

Floorplanning is an optional methodology that can be used to improve the performance and routability of your design. The objective in floorplanning is to assign logic to specific regions on the chip in order to enhance performance and routability.

When floorplanning, you analyze your design to see if certain logic can be clustered within regions. This is especially helpful for hierarchical designs with plenty of local connectivity within a block. If your timing analysis indicates several paths with negative slack, try clustering the logic included in these paths into their own regions. This forces the placement of logic within the path closer together and may improve timing.

Use ChipPlanner to help you floorplan. ChipPlanner can be used before and after Layout.

Regions

When floorplanning, you assign logic to regions to improve the design performance. Regions can be created using PDC or GCF files, or by using ChipPlanner. Spine regions can only be created using a PDC or GCF file.

Types of regions

There are four types of regions.

Region Type	Conditions
Empty	No macros can be assigned to an empty region.
Exclusive	Only contains macros assigned to the region. Not supported in ProASIC and ProASIC ^{PLUS} .
Inclusive	Can contain macros both assigned and unassigned to the region
Spine	Can either be defined as Exclusive or Inclusive in the PDC or GCF file. Cannot be resized. Can only contain certain types of macros.

Creating regions

Using ChipPlanner, you can create Empty, Exclusive, and Inclusive regions.

To create an empty or inclusive region:

1. From the ChipPlanner **Edit** menu, select **Regions**, and click **Create Empty** or **Create Logic**. Selecting **Create Logic** creates an inclusive region.
2. Drag the mouse over the area where you want the region to be placed.

To create an exclusive region:

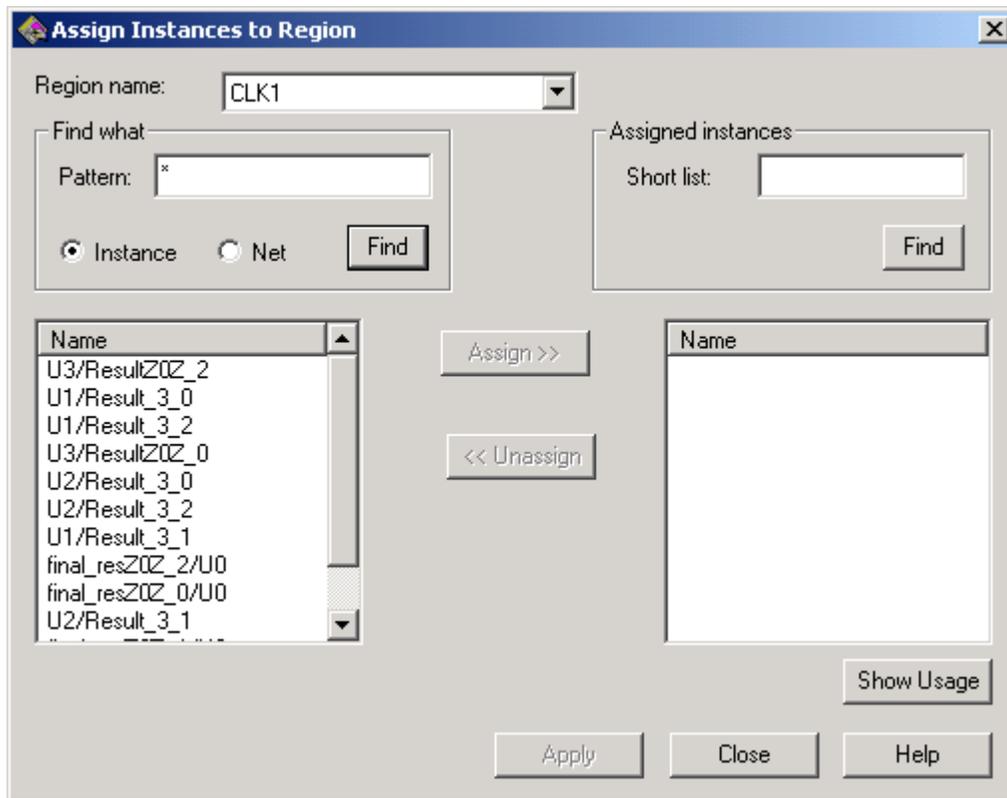
1. From the **Edit** menu, select **Regions**, and click **Create Logic**. Selecting **Create Logic** creates an inclusive region.
2. Drag the mouse over the area where you want the region to be placed.
3. Right-click the region and select **Properties**. The **Region Properties** dialog box appears.
4. Select **Exclusive** and click **OK**.

Assigning logic to regions

During floorplanning, logic can be assigned to regions to improve design performance.

To assign logic to regions:

1. Right-click a region and select **Assign/UnAssign**. The **Assign Instances to Region** dialog box appears.



Assign Instances to Region Dialog Box

2. Select **Instance**.
3. Enter a Pattern in the text box and click **Find**. To see all instances, type * and click **Find**.
4. Select the instance in the left list box and click **Assign**.

Tip: You can also assign logic to regions by using a drag-and-drop operation.

Assigning nets to regions

When assigning a net to a region, only the instances connected to the net are assigned to the region.

To assign nets to regions:

1. Right-click a region and select **Assign/UnAssign**. The Assign Instances to Region dialog box appears.

2. Select **Net**.
3. Enter a pattern in the Pattern text box and click **Find**.
4. Select the Net in the left list box and click **Assign**.

Editing regions

After creating regions with ChipPlanner, you can name, delete, move, and re-size them.

To name a region:

1. Right-click the region and select **Properties**. The Properties dialog box appears.
2. Type a new region name and click **OK**.

To delete a region:

Right-click on the region and select **Delete**.

To move a region:

Select and drag the region to a new location.

To re-size a region:

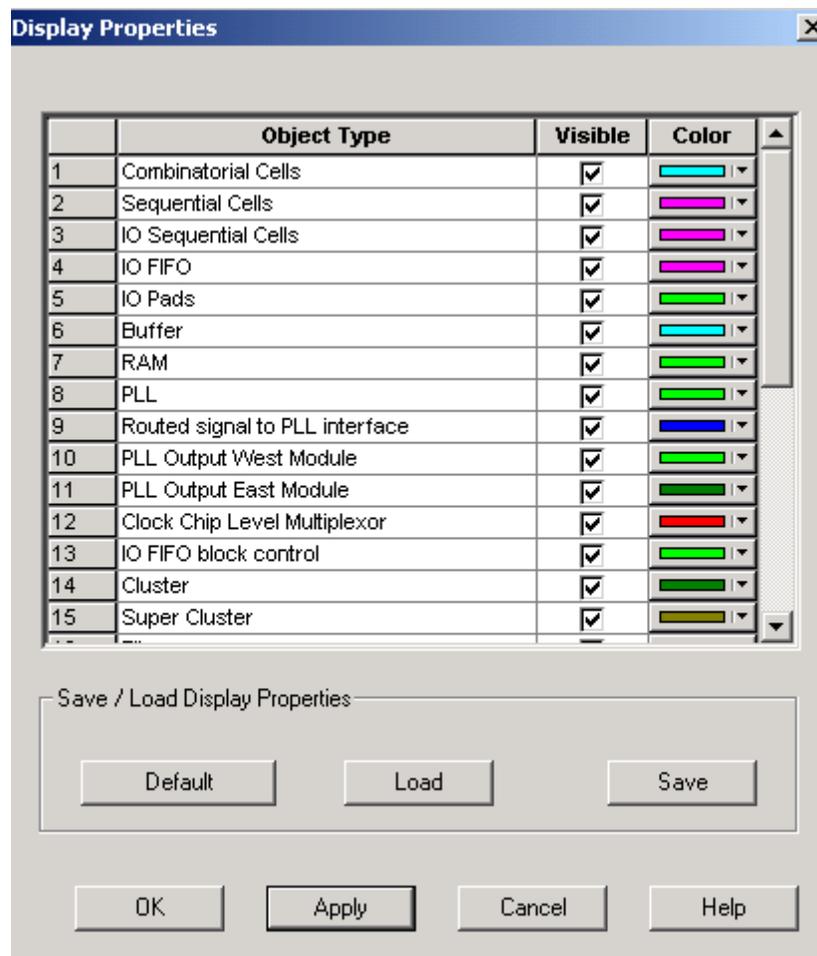
1. Select the region.
2. Grab and drag the sides and corners to re-size the region. A region cannot be re-sized smaller than the logic it already contains.

ChipPlanner Display Properties

To control what architectural features are displayed in ChipPlanner, use the Display Properties dialog box. For detailed information about supported architectural features, see the data sheet.

To set display properties:

1. From the **View** menu, click **Display Properties**.



Display Properties Dialog Box

All the architectural features that can be displayed appear in the Display Properties dialog box.

2. To make an object visible, select the Visible checkbox.
3. To change the color used to display the object, click the color bar and select another color.
4. **Save/Load Display Properties.**
 - Click **Save** to save your display properties to a file.
 - Click **Load** to open a saved display properties file.
 - Click **Default** to load the default display properties.
5. Click **Apply** to see your changes.
6. Click **OK** to dismiss the dialog box.

Ratsnest

The ratsnest view displays net connectivity between placed logic macros by connecting lines from the output pins to all input pins. Use the ratsnest to understand how logic macros are connected to each other. The ratsnest view is activated by default, showing all input and output nets for the selected macro.

Turn the Ratsnest view on or off by clicking the Ratsnest toolbar button.

Route view

The route view displays a representation of the actual routes used to connect placed macros. This feature helps show the general location of routing segments used by the design.

To activate the route view in ChipEditor:

1. Complete Layout. To display routes, Layout must be completed before running ChipEditor.

2. From the **Nets** menu, choose **input**, **output**, or **both** or click the corresponding Net toolbar icon.
3. From the **Nets** menu, choose **Display Algorithm Routes** or click the routes icon in the toolbar.
4. Select the placed macro in the ChipView window or Placed list box. Select multiple macros by holding down the CTRL key.

Note: If a macro is moved or unplaced, then the nets connected to that macro will be displayed using a ratsnest.

Net details

To display net details:

1. Select a net or locate the net by name.
2. From the **Nets** menu, click **Show Net Details**. The Net Details dialog box displays pin name and xy coordinates.

Clusters and SuperClusters

A cluster is a group of logic elements. The type of elements that make up the cluster is determined by the device type.

A super cluster is at least 2 clusters (SX) or 2 clusters and a buffer (Axcelerator). Modules in a cluster can be connected by fast or direct connects.

Use these areas as guides to ensure that the nets are fast/direct connect for implementation. Nets that connect within a rectangle can be implemented as fast or direct connects, depending on availability. For details about fast connects and direct connects, please see the Actel FPGA Databook.

Note: This feature is only available for the SX, SX-A, eX, and Axcelerator families.

To view clusters or super clusters in ChipPlanner:

1. From the **View** menu, click **Display Properties**. The Display Properties dialog box appears.
2. Locate cluster or super cluster and select the Visible check box. Click the color bar to change the display color.
3. Click **OK**.

To view clusters or super clusters in ChipEditor:

1. From the **View** menu, select **Static Objects** and click **Cluster Rect.** or **SuperCluster Rect.** The cluster areas appear in the ChipView window.

Finding Design Objects

Locating a net by name

To locate a net by name in ChipEditor:

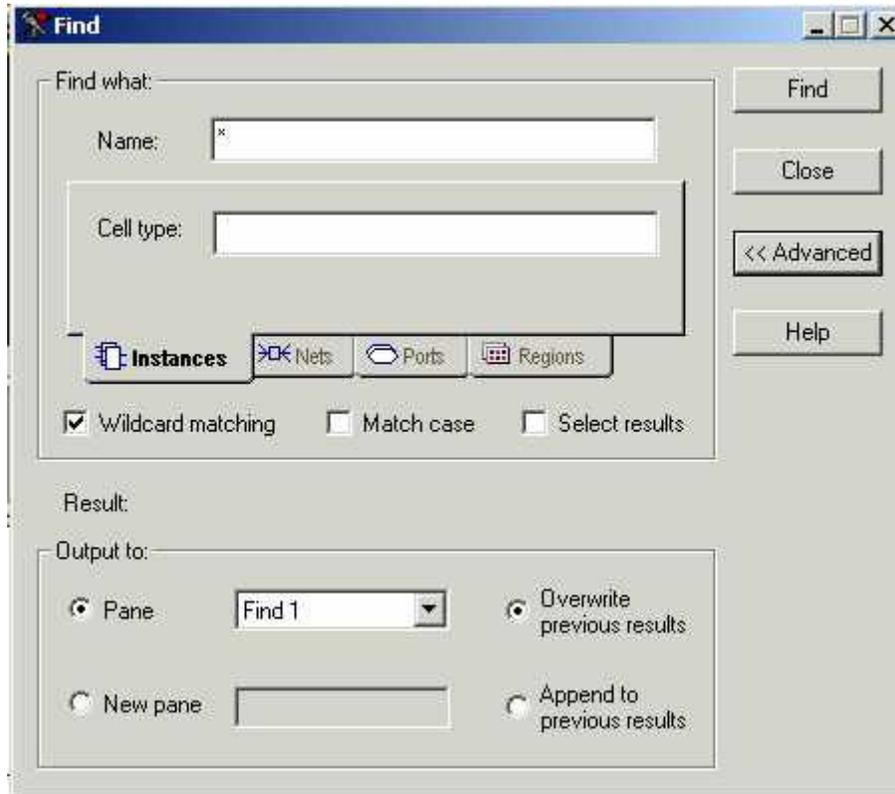
1. From the **Nets** menu, click **Select Net**.
2. Enter Net name and click **Find**. The net is highlighted in the ChipView window.

Finding objects

Use the Find feature in the MultiView Navigator to locate instances, nets, ports, and regions. You can use the Find feature when using any tool that opens in the MultiView Navigator interface.

To find instances:

1. From the **Edit** menu, click **Find**. The find dialog appears.
2. Click the **Instances** tab.



Find Instances Dialog Box

3. To search by name, type the name and Cell Type in the **Name** and **Cell Type** fields. When searching for instances, Instance Name or Cell Type can be blank, but not both.

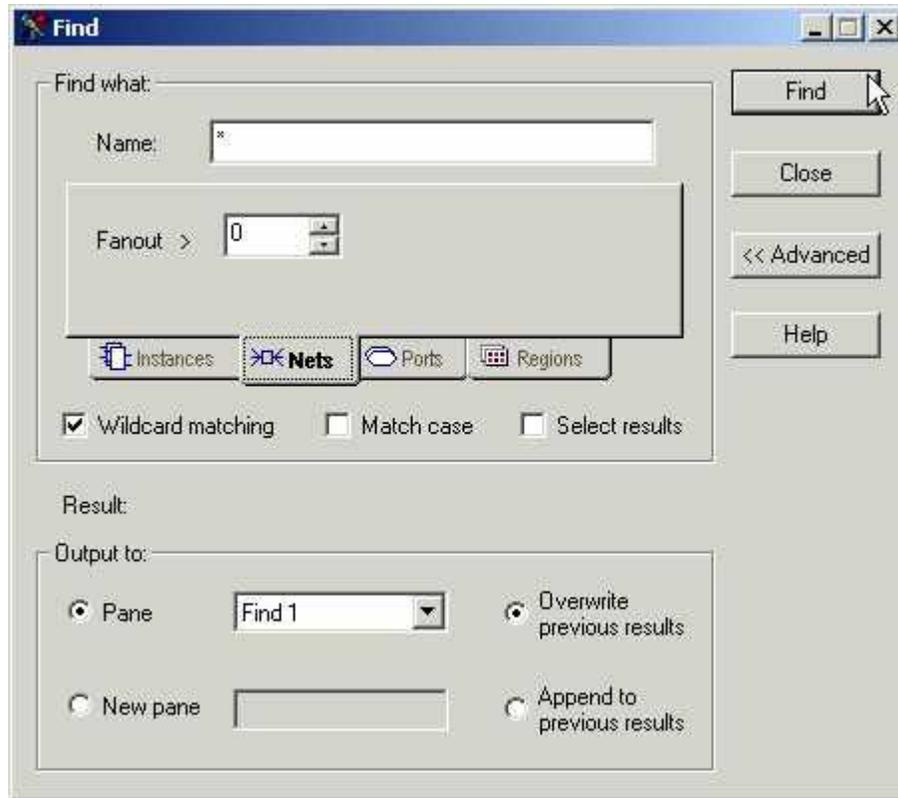
These fields accept regular expressions. Wildcards in regular expression include:

- ? matches any single character
- * matches any string
- [] matches any single character among those listed between brackets
- [A-Z] matches any single character in range A-Z

- [Z-A] matches any single character in range A-Z
 - / is the level-bordering symbol. "A/B" designates "object B, which is part of instance A". Note that the level-bordering symbol cannot be put between brackets in a regular expression.
4. Select **Wildcards** if you want to search using wildcards.
 5. Select **Match case** if you want the search to only return items with the exact characters specified.
 6. Click **Advance** to specify how you want your results displayed. Specify or create a new pane in the log window to display your results. If you use an existing pane, you can choose to overwrite your previous results or append the new results.
 7. Click **Find**. The located instances, if any, appear in the Find pane in the log window.

To find a net:

1. From the **Edit** menu, click **Find**. The find dialog appears.
2. Click the **Nets** tab.



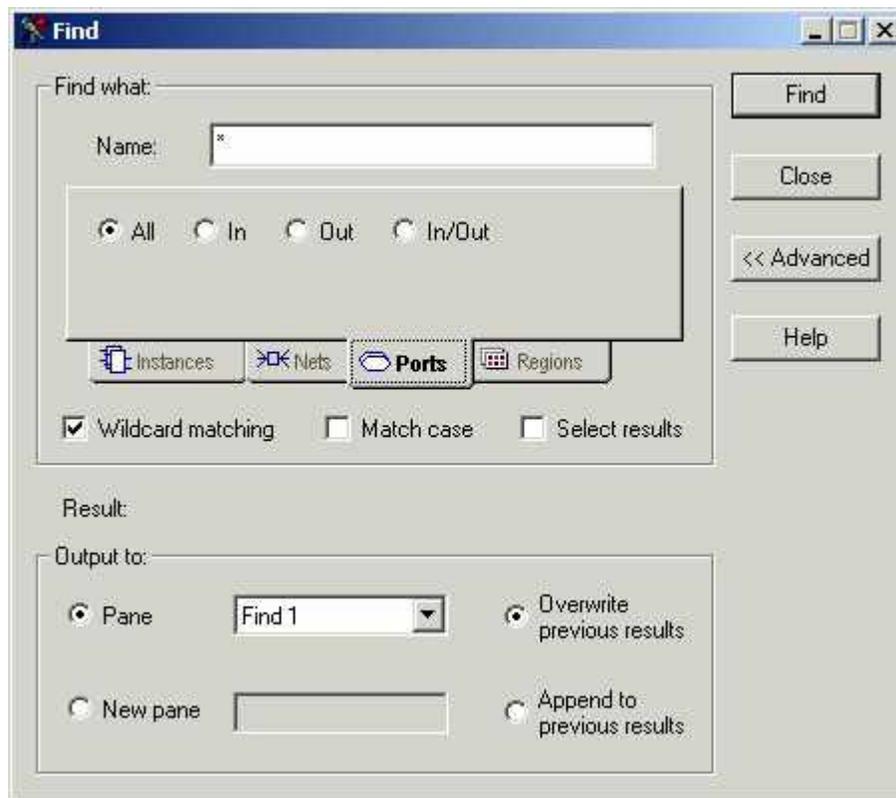
Find Nets Dialog Box

2. Type the **name** of the net. This field accepts regular expressions. Wildcards in regular expression include:
 - ? matches any single character
 - * matches any string
 - [] matches any single character among those listed between brackets
 - [A-Z] matches any single character in range A-Z
 - [Z-A] matches any single character in range A-Z
 - / is the level-bordering symbol. "A/B" designates "object B, which is part of instance A". Note that the level-bordering symbol cannot be put between brackets in a regular expression.
3. Select **Wildcards** if you want to search using wildcards.

4. Select **Match case** if you want the search to only return items with the exact characters specified.
5. Click **Advance** to specify how you want your results displayed. Specify or create a new pane in the log window to display your results. If you use an existing pane, you can choose to overwrite your previous results or append the new results.
6. Click **Find**. The located nets, if any, appear in the Find pane in the log window.

To find ports:

1. From the **Edit** menu, click **Find**. The find dialog appears.
2. Click the **Ports** tab.



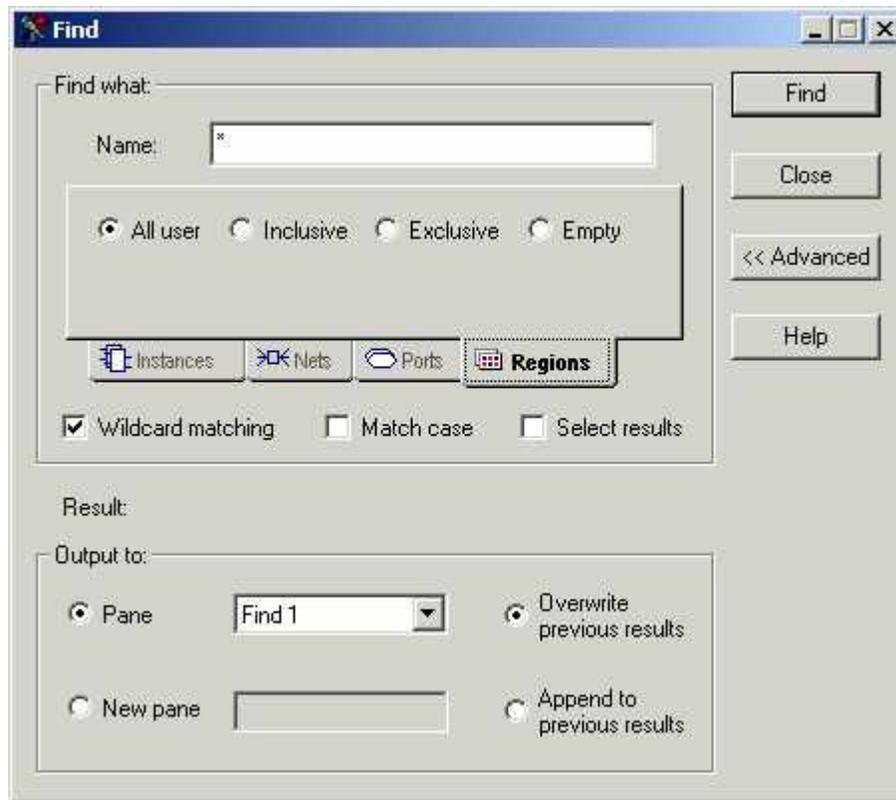
Find Ports Dialog Box

3. To search by name, type the **Name** of the port to be located. These fields accept regular expressions. Wildcards in regular expression include:

- `?` matches any single character
 - `*` matches any string
 - `[]` matches any single character among those listed between brackets
 - `[A-Z]` matches any single character in range A-Z
 - `[Z-A]` matches any single character in range A-Z
 - `/` is the level-bordering symbol. "A/B" designates "object B, which is part of instance A". Note that the level-bordering symbol cannot be put between brackets in a regular expression.
4. To find a port by type, select **All**, **In**, **Out**, or **In/Out**.
 5. Select **Wildcards** if you want to search using wildcards.
 6. Select **Match case** if you want the search to only return items with the exact characters specified.
 7. Click **Advance** to specify how you want your results displayed. Specify or create a new pane in the log window to display your results. If you use an existing pane, you can choose to overwrite your previous results or append the new results.
 8. Click **Find**. The located ports, if any, appear in the Find pane in the log window.

To find regions:

1. From the **Edit** menu, click **Find**. The Find dialog box appears.
2. Click **Regions**.



Find Regions Dialog Box

3. To search by name, type the **Name** of the region you want to find. These fields accept regular expressions. Wildcards in regular expression include:
 - ? matches any single character
 - * matches any string
 - [] matches any single character among those listed between brackets
 - [A-Z] matches any single character in range A-Z
 - [Z-A] matches any single character in range A-Z
 - / is the level-bordering symbol. "A/B" designates "object B, which is part of instance A". Note that the level-bordering symbol cannot be put between brackets in a regular expression.
4. To search by type of region, select **All User**, **Inclusive**, **Exclusive**, and **Empty**.

I/O Banks

For devices that support multiple I/O standards, I/Os are grouped onto I/O banks around the chip.

The Axcelerator Family has 8 I/O banks that surround the chip, two per-side, numbering 0-7. The I/O banks are color coded for quick identification. (Colors can be changed using the Color Manager.)

Each I/O bank has a common:

- VCCI, the supply voltage for its I/Os
- VREF, the reference voltage bus (for voltage-referenced I/O standards)

Only one VREF value can be assigned to each I/O bank. Only I/Os compatible with both the same VCCI and VREF standards can be assigned to the same bank.

Assigning technologies to I/O Banks

To assign technologies to banks:

1. Select an I/O bank.
2. From the **Edit** menu, click **I/O Bank Properties**.
3. In the Configure I/O Dialog Box, select your options and click **Apply**. The I/O bank is assigned the selected standards. Any I/O of the selected types can now be assigned to that I/O bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are removed.
4. If VREF pins can be assigned, the Assign VREF Pins button will highlight.

5. Assign I/O standards to other banks by selecting the banks from the list and assigning standards. Any banks not assigned I/O standards use the default standard selected in the Device Selection Wizard.
6. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

I/O Bank options

When assigning technologies to your I/O banks, use the Configure I/O Bank dialog box.

Options include:

Select Technologies

Selecting a standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Further selecting GTL (3.3V) disables SSTL3 as an option because the VREFs of the two are not the same.

Assign VREF Pins

After you have selected your technology, click **Apply**. If VREF pins are required, this button becomes activated. Click to assign VREF pins. You must assign VREF pins at least once.

Click More Attributes to set the following:

Low Power Mode (Optional)

Select **Enable Input Buffers** or **Enable Output Buffers**. These are not required. This feature is not supported in the RTAX-S family.

Input Delay

Drag the slider bar to your desired delay. The delay is bank specific. Drag the meter to your desired delay index. The delay code and typical value appear. Click **View All Delays** to see all the delay values (Best, Worst, Typical, Rise-Rise, Fall-Fall) for the input delay selected. A technology must be selected in order to see the input delays. Click **OK** to dismiss the View All Delays dialog box. This feature is not supported in the RTAX-S family.

Assigning VREF pins

Voltage referenced I/O inputs require an input referenced voltage (VREF).

To assign VREF pins:

1. From the **Edit** menu, click **Assign I/O Technologies to I/O Banks**.
2. Specify the supported technologies for the I/O bank and click **OK**.
3. If VREF pins can be assigned, the Assign VREF Pins button activates.
4. Click **Assign VREF Pins**. The Assign VREF Pins dialog box appear
5. Check the VREF box next to the pin number and click **OK**. Click the **Reset to Defaults** button to revert to Actel recommended defaults.
6. Click **OK** to dismiss the Assign I/O Technologies to I/O Banks dialog box.

Specifying I/O bank voltage

You can directly specify voltages for each I/O bank by doing one of the following:

- Using the Assign Technologies to I/O Banks dialog box
- Placing an I/O of a particular technology in an I/O bank that has not been assigned a voltage
- Using the command `set_iobank` in a PDC File

Using ChipEditor with Silicon Explorer

Use ChipEditor to select probes for Silicon Explorer. To use ChipEditor with Silicon

Explorer, you must have installed and be familiar with Silicon Explorer.

To select probes using ChipEditor:

1. Open Silicon Explorer.
2. Load the probe file of the current design.
3. Start ChipEditor.
4. Synchronize data. Click the R (Re-sync) button in ChipEditor's toolbar. When completed, the A and B buttons in the toolbar are activated.
5. Select a module in ChipEditor.
6. Click the A button in ChipEditor's toolbar to assign the selected module's output to probe A in Silicon Explorer.
7. Select another module in ChipEditor.
8. Click the B button in ChipEditor's toolbar to assign the selected module's output to probe B in Silicon Explorer.
9. From Silicon Explorer, click the Acquire toolbar button. Waveforms are displayed in Silicon Explorer.

Using ChipEditor with Timer

Use ChipEditor and Timer together to view place-and-route of paths in ChipEditor.

To view paths:

1. Open Timer and ChipEditor from Designer.
2. In **Timer**, click the **Paths** tab.
3. Select a Path set in the path set grid. Paths within that set are displayed below in the path grid.
4. Select the path you wish to expand in the lower path grid.

5. Expand the path by double-clicking on the path, or in the **Edit** menu, click **Expand Path**. The Expanded Paths window opens and displays a path in the Expanded Paths Grid and a graphical representation of the path in the Chart Window. The Expanded Paths grid shows all delay components for the selected path (Instance, Net, Macro, Delay, Type, Total Delay and Fanout details). For Delay, (r) stands for rising edge and (f) for falling edge.
6. Anything selected in the Expanded Paths grid or Graph window is reflected in both windows. Selecting the path number in the Expanded Paths grid highlights the entire path in the Chart window.
 - Selecting an instance, net, or macro in the Expanded Paths grid highlights that selection in the Chart window.
 - Selecting a logic macro in the Chart window, highlights all instances of the macro in the Expanded Paths grid.
 - Toggle the Graph Window on and off by clicking Graph Window from the Window menu. Use the View command menu to Zoom in and out. In the Graph window, dragging the mouse downward and to the left will zoom fit. Dragging downward and to the right drags out a zoom in area.
 - In some cases, long instance names may overlap and be difficult to read in the Graph window. This problem can be solved by moving the module. To move the module, select the module and while holding down the Shift key, click and drag the module to another location.
7. Select a module or net in the Expanded Paths dialog box. The module or net is shown in ChipEditor.

Using the ChipEditor Interface

ChipEditor consists of the ChipView window, the Package window, and the Placed and Unplaced list boxes.

ChipView Window

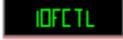
The ChipEditor ChipView window displays logic modules and placed macros. When you select a macro in the ChipView window, the macro location is highlighted in the World View window and the macro name is selected in the Placed list box.

To zoom in and out, use the commands in the View menu or toolbar. To use hot keys, place your mouse over the desired zoom area and use Shift and the "+" plus key to zoom in on the location and Shift and the "-" minus key to zoom out.

Colors and Symbols Used in ChipEditor

Colors and symbols are used to differentiate the I/O and logic macros in the ChipView Window.

Chip Window Colors and Symbols Color/Symbol	Definition
White Border	A white border denotes a selected object.
Black Background	A black background denotes an unused or unplaced module.
Blue	Blue denotes a combinatorial module.
<i>Yellow</i>	Yellow denotes <i>fixed</i> logic modules. If the module is selected, the symbol appears yellow. If the module is unselected, the border appears yellow.
Green	Green denotes I/O modules.
Red	Red denotes clock modules.
Magenta	Magenta denotes sequential modules.
	Reserved modules that are not user definable are gray, crossed-out symbols on a black background.
	Clock modules are red. Unused/unplaced modules are red symbols on a black background. Used/placed modules are black symbols on a red background.
	Input/Output modules <i>are</i> green. Unused/unplaced modules are green symbols on a black background. Used/placed modules are black symbols on a green

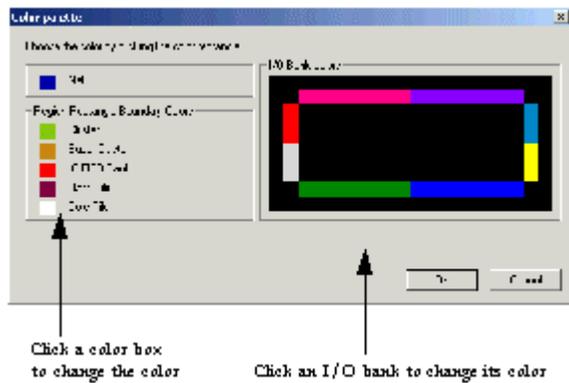
	background.
	Combinatorial modules are blue. Unused/unplaced modules are blue symbols on a black background. Used/placed modules are black symbols on a blue background.
	Sequential modules are magenta. Unused/unplaced modules are magenta symbols on a black background. Used/placed modules are black symbols on a magenta background.
	Buffer modules are blue.
	RAM modules are green. Unused/unplaced modules are green symbols (RAM) on a black background. Used/placed modules are black on a green background.
	PLL modules are green. Unused/unplaced modules are green symbols (PLL) on a black background. Used/placed modules are black on a green background.
	I/O FIFO Block Controller modules are green. Unused/unplaced modules are green symbols (IOFCTL) on black backgrounds. A used/placed module is black on a green background.
	I/O FIFO Inbuff modules are pink on a black background. Used/placed modules are black on a pink background.
	I/O Inbuff modules are pink on a black background. Used/placed modules are black on a pink background.

Changing Colors in ChipEditor

Customize the colors used to display I/O banks, clusters, SuperClusters, and nets in the ChipView window by using the Color Manager.

To customize colors in the ChipView window:

1. From the **View** menu, click **Color Manager**.



2. Click the color box in front of the item you wish to customize, or click the I/O bank you wish to change to see the color pallet.



3. Select a new color and click **OK**. The new color appears in the Color Manager dialog box.
4. When you are done customizing your colors using the Color Manager dialog box, click **OK**.

Placed and Unplaced List Boxes

The ChipEditor Placed and Unplaced list boxes display placed or unplaced macros in the design. All placed macros appear in the Placed list box and all unplaced macros appear in the Unplaced list box.

To configure the list boxes:

1. From the **View** menu, click **Configure List Boxes**.
2. In the Configure List Boxes dialog box, select from the following options:
 - **Filter Placed and Unplaced Lists:** Entering a macro name in this area will filter for a specific macro or group to be displayed in the Placed or Unplaced list box. You can use the "*" character as a wildcard.
 - **Placed List Box Filters:** Use these radio buttons to filter the Placed List Box to display fixed and unfixed macros, only fixed macros, or only unfixed macros.
 - **Unplaced List Box Filters:** Use these radio buttons to display all macros or just those that must be manually placed.
 - **List Type:** Use the List Type filters to display macro instance names in a flat or hierarchical list in the Placed and Unplaced list boxes. When instance names are displayed hierarchically, collapsed levels are preceded by a plus sign (+) and expanded levels are preceded by a minus sign (-). Clicking the plus sign expands the hierarchy of a macro, while clicking the minus sign collapses the hierarchy. Macros, both fixed and unfixed, are displayed hierarchically by default.

World View Window

Use the World View window to control which portion of the ChipView is displayed in the ChipView window. The blue rectangle (known as the ChipView rectangle) represents the chip. The green rectangle (known as the Viewing rectangle) represents the area displayed in the ChipView window.

To move the displayed area to another part of the chip, click the left mouse button and drag the Viewing rectangle to the area on the ChipView rectangle you would like to display. To specify a new display area, click the right mouse button and drag-out a new Viewing rectangle on the ChipView rectangle.

ChipEditor Status bar

Family, die and package information appears in the right corner of the status bar. In addition, the status bar displays information on commands, pins, placed macros, nets, error messages, and the family, die, and package.

- Hold your mouse over a placed macro in the ChipView window to see the pin number, instance name, net name, macro cell, and fixed or unfixed status in the Status Bar.
- To see nets displayed in the status line, select a macro, zoom in, and click one of the ratsnest lines.
- If you hold your mouse over a toolbar icon or a menu command, a short description of the command function appears in the Status Bar.
-

Error messages in the Status Bar provide details about invalid placement attempts.

Choose

Help > Extended Error Messages to view more information about the last failed command or placement attempt.

Using PDC and GCF files

About Physical Design Constraint (PDC) Files

A PDC file is a Tcl script file specifying physical constraints. This file can be imported and exported from Designer. Any constraint that can be entered by using PinEditor, ChipEditor, or ChipPlanner can also be specified in a PDC file. In addition, a PDC file allows you to specify region constraints and net criticalities.

Note: PDC files are only supported for the Axcelerator family of devices. Use a PDC file

instead of a PIN file.

Supported commands include:

Command	Action
assign_net_macros	Assigns the macros connected to a net to a specified defined region
assign_region	Assigns macros to a pre-specified region
define_region	Defines a rectilinear region
define_region	Defines a rectangular region
reset_floorplan	Deletes all defined regions. Placed macros are not affected
reset_iobank	Resets an I/O banks technology to the default technology
reset_io	Resets all attributes on a macro to the default values
reset_net_critical	Resets net criticality to default level
set_io	Sets I/O attributes
set_iobank	Specifies the I/O bank's technology
set_location	Places a given logic instance at a particular location
set_net_critical	Sets net criticality, which is issued to influence placement and routing in favor of performance
set_vref	Specifies which pins are Vref pins
set_vref_defaults	Sets the default vref pins for specified banks
unassign_macro_from_region	Unassigns macros from a specified region, if they are assigned to that region
unassign_net_macro	Unassigns macros connected to a specified net from a defined region
undefine_region	Undefines the region

About GCF Files

A GCF file is an ASCII file specifying design constraints. This file can be imported and exported from Designer.

Note: GCF files are only supported for the Flash families.

Importing PDC files (Axcelerator family only)

The Physical Design Constraint (PDC) file can specify:

- I/O standards and features
- VCCI and VREF for all or some of the banks
- Pin assignments
- Placement locations
- Net criticality

The Axcelerator family of devices supports multiple I/O standards (with different I/O voltages) in a single die. You can use ChipEditor and PinEditor to set I/O standards and attributes, or alternatively you can export and import this information using a PDC file. PDC files are only supported for the Axcelerator family of devices.

To import a PDC file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog appears.
2. Click the **Add** button. The Add Auxiliary Files dialog box appears. Filter for your PDC file by selecting Physical Design Constraint Files (*.pdc) from the Files of Type drop-down list box.

3. Select the PDC file and click **Import**. The file is added to the Import Auxiliary Files dialog box.
4. Click **OK**. The PDC file is imported into Designer. Any errors appear in the Log Window.

Note:

- File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.
- If the PDC file has commands to combine I/O Registers with I/Os this file must be imported before compile

Exporting PDC files

PDC files can be exported from Designer.

To export a PDC file:

1. From the **File** menu, click **Export, Constraint File**.
2. Type a file name and choose a directory. Click **OK**. The Export Physical Design Constraints (PDC) dialog box appears.
3. Choose the type of information you want to export:
 - **Pin locations and attributes:** Select to export information about the pin locations and attributes only.
 - **Placement constraints:** Select to export information about the I/O placement and routing constraints only.
 - **Complete placement information:** Select to export information about the I/O locations, I/O attributes, placement, and routing constraints.
4. Click **OK**.

Importing GCF files

Import GCF files as you would any source file.

To import a GCF file:

1. In the **File** menu, click **Import Source Files**.
2. Click **Add**. The Add Source Files dialog appears.
3. Select ProASIC Constraint Files (.gcf) from Files of type.
4. Select your .gcf file and click **Import**. The File is added to the Import Source Files dialog box.
5. Add any more source files to the list. All files added to the Import Source Files dialog box are imported at the same time.
 - **Modifying:**If you need to modify a selection, select the file row and click **Modify**
 - **Deleting:**If you need to delete a file, select the file row and click **Delete**.
 - **Ordering:**Ordering your source files. Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files. When importing multiple EDIF or structural HDL files, the top-level file must appear last in the list (at the bottom).
6. After you are done adding all your source files, click **OK**. Your source files are imported. Any errors appear in the Designer Log Window.

Exporting GCF files

To export a GCF file:

1. From the **File** menu, click **Export, Constraint Files**.
2. Enter the file name and click **OK**.

3. Chose the type of information you want exported:
 - **Pin locations:** Select to export the I/O placement and region constraints related to the I/Os only.
 - **Placement constraints:** Select to export all placement constraints, including I/O and core constraints.
 - **All GCF constraints:** Select to export all constraint information.
4. Click OK.

Types of Constraints

Constraints are used to ensure that a design meets timing performance and required pin assignments. For Flash families, the types of constraints that can be defined in a .gcf constraint file include:

- Timing constraints
- Global resource constraints
- Netlist optimization constraints
- Placement constraints

Constraint File Syntax

A ProASIC constraint consists of a statement and an argument, terminated by a semicolon. Statements are not case sensitive. However, cell instance, net, and port names used as arguments may be quoted and are case sensitive. Except for white spaces, all ASCII characters can be used. Comments are allowed in constraints files and must be preceded by two forward slashes (//). Time values are given in nanoseconds. When constraints are duplicated, the last one specified for a specific item overwrites any previous similar constraints already specified for the considered item.

Constraint Quick Reference

- create_clock

- generate_paths
- set_input_to_register_delay
- net_critical_ports
- set_critical
- set_critical_port
- set_max_path_delay
- set_switch_threshold
- set_auto_global
- set_global
- dont_fix_globals
- use_global
- dont_optimize
- optimize
- set_net_region
- set_max_fanout
- dont_touch
- set_empty_location
- set_empty_io
- set_initial_io
- set_io
- set_location
- set_initial_location

GCF Syntax Conventions

This section describes syntax conventions for notation, user data variables, and comments. Comments begin with double slashes (//) and are terminated by a newline character.

Syntax Conventions for Notation

Notation	Description
item	Represents a syntax item.
item ::= definition	item is defined as definition.
item ::= definition1 = definition2	item is defined as either definition1 or definition2. (Multiple alternative syntax definitions are allowed.)
[item]	item is optional.
{ item }	item is a list of required items. At least one item must appear.
KEYWORD	Keywords appear in uppercase characters in bold type for easy identification, but are not case sensitive.
VARIABLE	Represents a variable and appears in uppercase characters for easy identification.

Syntax Conventions for User Data Variables

Variable	Description
-----------------	--------------------

FILEIDENTIFIER	Represents a hierarchical filename.
IDENTIFIER	Represents the name of a design object. Can be a block, cell instance, net, or port. IDENTIFIERS can use any ASCII character except the white space and the slash (/), which is the hierarchical divider character (see QPATH below). IDENTIFIERS are case sensitive.
POSFLOAT	Represents a positive real number; for example, 4.3, 1.15, 2.35.
POSNUMBER	Represents a positive integer; for example, 1, 12, 140, 64. When representing time, POSNUMBER is expressed in nanoseconds (ns).
QPATH	Represents a hierarchical IDENTIFIER. The levels of the hierarchy are represented by IDENTIFIERS divided by a slash (/). The QPATH hierarchical IDENTIFIER may or may not be quoted.

About Global Resource Constraints

Each ProASIC and ProASIC^{PLUS} device includes four global networks that have access to every tile. These four global networks provide high speed, low skew performance to signals such as clocks and global reset.

Once the netlist is imported, Designer sets global resource parameters and promotes the highest fanout nets to the remaining global resources unless the “dont_fix_globals” statement has been specified in a constraint file. To do this, the importer program demotes appropriate global cell instantiations in the design netlist.

Note: When using the “dont_fix_globals” statement, global assignments made in the constraint files and design netlist will be honored (the constraint file entries will take precedence).

These global resource parameters can be supplemented by including globalg resource constraints in a constraint file. Global resource constraints can define which signals are assigned to global resources and which signals cannot be promoted to global resources. Global resource constraints can also override the default action that selects high fanout nets for use by the global resources. If global resources overrides the default action, assignments that do not include any of the four highest fanout nets will generate a warning.

Priority Order for Global Promotion

While assigning signals to global resources, Designer considers this information in the given priority:

1. set_global and set_io statements (instances of those global cells, which cannot be demoted)
2. Nets with the highest potential fanout above 32 (after removal of all buffers and inverters)
3. Global cell instantiation in a netlist (global cells which can be demoted)

Note:

- By default, a net with a fanout of less than 32 will not be promoted to global automatically unless the “set_global” or “set_io” constraint statements is used for this net. Users can override this threshold of 32 by using the “set_auto_global_fanout” constraint statement.

dont_fix_globals

Use this statement to turn off the default action that automatically corrects the choice of global assignment to use only the highest fanout nets.

```
dont_fix_globals;
```

set_auto_global

Use this statement to specify the maximum number of global resources to be used. The tool assigns global resources to high fanout signals automatically.

If the user specifies a number that exceeds the actual number of global resources available in the device, Designer ignores the statement. If the user specifies 0, no automatic assignment to global resources will take place.

```
set_auto_global number ;
```

For example, the following statement specifies that of the possible four global nets available, the tool can automatically promote only two high fanout nets:

```
set_auto_global 2;
```

set_auto_global_fanout

Use this statement to set the minimum fanout a net must have to be considered for automatic promotion to a global. By default this is set to 32.

```
set_auto_global_fanout number ;
```

For example, the following statement determines that a net must have at least a fanout of 12 before Designer considers it for automatic promotion to a global resource.

```
set_auto_global_fanout 12;
```

set_global

Use this statement to classify nets as global nets.

```
set_global hier_net_name [ , hier_net_name ... ];
```

For example:

```
set_global u1/u3/net_clk, u3/u1/net_7;
```

set_noglobal

Use this statement for classifying nets to avoid automatic promotion to global nets.

```
set_noglobal hier_net_name [ , hier_net_name ... ];
```

For example:

```
set_noglobal u2/u8/net_14;
```

If the net was previously assigned to a global resource, this statement will demote it from the global resource.

use_global

This statement allows you to specify a single spine or a rectangle of spine region which may encompass more than 1 spine region.

```
use_global T2 <net_name>;  
use_global B1, T3 <net_name>;
```

For example, if you give the spine rectangle as B1, T3. The driven instances of the given net get a region constraint which encloses the rectangle including the spine rectangle B1, T1, B2, T2, B2, T3.

It tries to place the driver as close to center of the rectangle as possible.

You can specify the following type of rectangles:

1. $B_n, B_m : n \leq m$ will mean $B_n, B_{n+1}, \dots B_m$
2. $T_n, T_m : n \leq m$ will mean $T_n, T_{n+1}, \dots T_m$
3. $B_n, T_m : n \leq m$ will mean $B_n, T_n, B_{n+1}, T_{n+1} \dots B_m, T_m$

4. $T_n, B_m : n \leq m$ will mean $B_n, T_n, B_{n+1}, T_{n+1} \dots B_m, T_m$

See table for a summary of available spines.

Global Spine Usage

Device	Spine
A500K050	T1 to T3
	B1 to B3
A500K130	T1 to T5
	B1 to B5
A500K180	T1 to T6
	B1 to B6
A500K270	T1 to T7
	B1 to B7
APA075	T1 to T3
	B1 to B3
APA150	T1 to T4
	B1 to B4
APA300	T1 to T4
	B1 to B4
APA450	T1 to T6
	B1 to B4
APA600	T1 to T7

	B1 to B7
APA750	T1 to T8
	B1 to B8
APA1000	T1 to T11
	B1 to B11

Note that T1 and B1 are the leftmost top and bottom global spines, respectively.

Netlist Optimization Constraints

Netlist optimization attempts to remove all cells from a netlist that have no effect on the functional behavior of the circuit. This reduces the overall size of a design and produces faster place and route times. This optimization is based on the propagation of constants and inverter pushing and takes advantage of inverted inputs of the basic logic elements. Refer to the *ProASIC 500K Family* and *ProASIC PLUS* Data Sheet for detailed information.

Netlist optimization can be controlled by including netlist optimization constraints in constraint files submitted to Designer.

By default all optimizations will be performed on the netlist. To control the amount of optimization that takes place, netlist optimization constraints can be used. Netlist optimization constraints can turn off all optimizations or disable the default action that allows all optimizations to limit the type of optimizations performed. The constraints can also define a maximum fanout to be allowed after optimizations are performed and isolate particular instances and hierarchical blocks from the effect of optimization.

After completion of netlist optimization, the design is a functionally identical representation of the design produced internally for use by Designer. View the design's layout after successful placement and routing. After optimization, a number of instances that do not contribute to the functionality of the design may have been removed.

To keep the SDF file consistent with the original input netlist, deleted cells are written with zero delay so that backannotation is performed transparently.

Netlist Optimization Constraint Syntax

The following netlist optimization options are available for all netlist optimization constraints.

- **buffer** - removes all buffers in the design provided that the maximum fanout is not exceeded.
- **const** - replaces all logical elements with one or more inputs connected to a constant (logical "1" or "0") by the appropriate logic function. If the replacement logic function is identified as an inverter or buffer, that element is removed.
- **dangling** - recursively removes all cells driving unconnected nets.
- **inverter** - removes all inverters in the design provided that the maximum fanout is not exceeded.

dont_optimize

This statement turns off all netlist optimizations. When followed by one or more of the netlist optimization options, this statement turns off the named optimization option. Buffers and inverters coming in the path of the global nets or spine nets are removed.

```
dont_optimize [{ inverter buffer const dangling}];
```

dont_touch

This statement allows the user to selectively disable optimization of named hierarchical instances. The wildcard "*" can be used to isolate all sub-blocks under the named block.

```
dont_touch hier_net_name [ , hier_net_name ... ];
```

For example:

```
dont_touch /U1/myblock/*;
```

Optimization types and optimization are done on all instances except those contained in the block called /U1/myblock.

optimize

This statement turns on all netlist optimizations (the default mode). When followed by one or more of the netlist optimization types, this statement enables only the named optimization(s).

```
optimize [{ inverter buffer const dangling} ];
```

For example:

```
optimize buffer inverter;
```

set_max_fanout

Use this statement to specify the maxFanout limit on the specific nets. Use when optimizing the buffers and inverters. The buffers and inverters are not removed if the fanout for the given net exceeds the given limit. If no net name is given, then the command is applied to all the nets in the design. The net name can be a simple net or a name having wild card characters.

```
set_max_fanout NUMBER <net_name_wildcard>;
```

Placement Constraints

It is possible to use placement constraints to specify block-instance and macro placement. Users can specify initial, fixed, region, and macro placements. Also, placement obstructions (locations that are not to be used and thus to be keep empty during placement instances) can be specified.

For example, a constraint that places two connected blocks close together usually improves the timing performance for those blocks. Similarly, a constraint that assigns an I/O pin to a particular net forces the router to make the connection between the driving or receiving cell and the I/O itself.

Like all constraints, placement constraints limit Designer's freedom when processing the design. For instance, assigning a fixed location makes that location unavailable during placement optimization. Such removal usually limits the program's ability to produce a chip-wide solution.

Macro

```
macro name (x1, y1 x2, y2) {  
macro_statements  
}
```

Where *name* is the macro name identifier, x1, y1 is the lower left coordinates of the macro, and x2, y2 is the upper right coordinates of the macro.

For example:

```
macro mult (1,1 6,6) {  
set_location...  
}
```

Now you can use the “set_location” or “set_initial_location” statements to place or initially place a sub-design instance by calling its macro and then applying a translation and rotation. This statement has been extended to allow you to initially place a sub-design instance by calling its macro and then applying a translation and rotation.

```
set_initial_location (x, y) hier_subdesign_inst_name  
macro_name [transformations];
```

For example:

```
set_location (3,3) a/b mult flip lr;
```

Where “*hier_subdesign_inst_name*” is the hierarchical name of the instance of the sub-design, *x, y* is the final location of the lower left corner of the macro after all transformations have been completed, *macro_name* - is the name of previously defined macro, and transformations are optional and any of the following in any order:

- **flip lr** - flip cell from left to right
- **flip ud** - flip cell from up to down
- **rotate 90 cw** - rotate 90° clockwise
- **rotate 270 cw** - rotate 270° clockwise
- **rotate 90 ccw** - rotate 90° counter-clockwise
- **rotate 180 ccw** - rotate 180° counter-clockwise
- **rotate 270 ccw** - rotate 270° counter-clockwise

The transformations are processed in the order in which they are defined in the statement.

For example:

```
set_initial_location (3,3) a/b mult flip lr;
```

Package Pin and Pad Location

Generally, you are concerned with the mapping of signals (ports) to the pins of the selected package. However, you may want to control the allocation of signals to particular pads. This is accomplished by assigning ports to the pad location rather than to the package pin. Because all pads are pre-bonded to package pins, the effect is to assign ports

to package pins, with the emphasis on pad location rather than package pin.

Pad location is described by the letters N (North), S (South), E (East) or W (West) followed by a space and a number. This location code determines the direction and offset of the pad with respect to the die.

The top edge of the viewer contains the North pads and the right edge contains the East pads. The number refers to the pad position along its edge. For example, N 48 corresponds to the 48th pad along the North edge of the die. Figure 4-2 on page 78 shows the numbering system used for pad location.

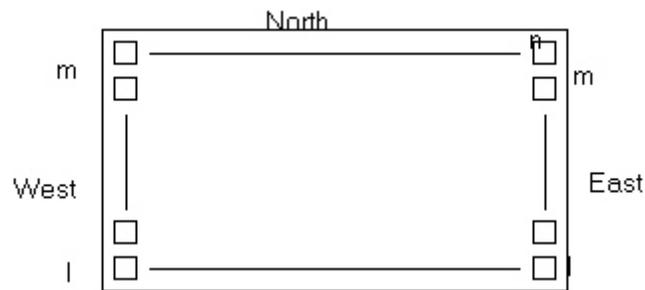


Figure A-1. Pad Locations

set_empty_io

Use this statement to specify a location in which no I/O pin should be placed. The location can be specified by side and offset or by name.

```
set_empty_io { package_pin | pad_location};
```

For example, the following statement forces pin B5 and the pin associated with the fourth tile on the North side to be empty:

```
set_empty_io B5, (N,4);
```

set_empty_location

Use this statement to specify a location in which no cell should be placed.

```
set_empty_location ( x ,y);
```

```
set_empty_location (xbl ,ybl xtr ,ytr );
```

Where x , y (required) are the (x, y) tile coordinates that specify the empty cell location and x_{bl} , y_{bl} x_{tr} , y_{tr} (required) are the x, y tile coordinates for the bottom left and top right corner of the region.

Note: Only white spaces are allowed between the coordinates.

For example, the following statement informs the placement program that location (3, 7) is unavailable for cell placement:

```
set_empty_location(3,7);  
set_empty_location(113,160,80,50);
```

set_initial_io

Use this statement to initially assign package pins to I/O ports or locate I/O ports at a specified side of a device. The placer can reassign or relocate the cells during placement and routing.

```
set_initial_io { package_pin | pad_location } io_port_name  
[ , io_port_name , ... ];
```

Where “package_pin” is a package pin number for a specified I/O cell.

If you use “package_pin,” only one “io_port_name” argument is allowed (required if no pin location is given). “pad_location” is one of N, S, E, or W, followed by a pad location number on the chip. It constrains the pin location of a specified I/O cell to a specific pad location on the chip. Only one “io_port_name” argument is allowed. “io_port_name” (required) is the name of an I/O port to be assigned to a package pin or located at a specified edge of a package.

The following example statement initially places the I/O associated with net in3 to package pin A11:

```
set_initial_io All in3;
```

The following example statement initially places the I/O associated with net in4 on the fourth tile on the

North side:

```
set_initial_io (N,4) in4;  
set_initial_io to a side is missing.
```

For example,
set_initial_io S in5; // assigns in5 to the South side

Multiple comma-separated ports can be specified when they are assigned to a side.

set_initial_location

Use this statement to initially locate a cell instance at specified x, y coordinates. The placer can relocate the cell instance during place and route.

```
set_initial_location ( x, y ) hier_inst_name ;
```

Where x, y (required) are the x, y tile coordinates for the location of a specified cell instance and “hier_inst_name” (required) is the hierarchical path to a cell instance.

For example:

```
set_initial_location (43,105) bk3/fp5/nand3_4;
```

set_io

Use this statement to either assign package pins to I/O ports or locate I/O ports at a specified side or location of a device. This constraint is a hard constraint and can not be overruled by the placer. This may have an impact on the timing results of a design. If a hard constraint is not suitable, use the “set_initial_io” constraint.

```
set_io {...|...} netName/portName;
```

For example:

```
set_io A9 in1;
set_io (S,22) in2;

set_io N in3;    // assigns in3 to the North side
```

Multiple comma-separated ports can be specified when they are assigned to a side.

set_location

Use this statement to locate a cell instance at specified x,y coordinates. The placer cannot relocate the cell instance during place and route.

```
set_location (x, y) hier_inst_name ;
set_location (x bl, y bl x tr, y tr) hier_inst_name/*;
```

Where x , y (required) are the (x, y) tile coordinates that specify the empty cell location and x bl , y bl x tr , y tr (required) are the x, y tile coordinates for the bottom left and top right corner of the region.

For example:

```
set_location (1,15) u4/u3/nand3_4;
set_location (1,1 32,32) datapath/*;
```

set_net_region

This GCF constraint enables you to put all the connected instances, driver and all the driven instances, for the net(s) into the target rectangle specified in the constraint. It puts the region constraint on all the connected instances, which will be processed by the placer. The IOs are excluded from the region constraints. The net name can take wildcards.

```
set_net_region (x1, y2 x2, y2) <net_name_wildcard>;
```

Note:

- Only white spaces are allowed between the coordinates.

Tcl Documentation Conventions

The Actel command syntax conventions are as follows.

Syntax Notation	Description
command	Commands or keywords are shown in <code>courier</code> typeface.
<i>Variable</i>	Variables appear in italic. You must substitute an appropriate value for the variable.
?argument?	Optional argument. Do not use the question marks when entering the argument.
arg1 arg2 ... argN	Alternative arguments. You can use exactly one of these arguments.
...	The ellipsis indicate items that precede the ellipsis may be repeated. The ellipsis should not be entered.

assign_net_macros

The `assign_net_macros` command assigns macros connected to a net to a region.

Syntax

```
assign_net_macros
  region_name net1 ?netN?...
```

Arguments

region_name

Since the `define_region` returns a region object, users can write a simpler command such as:

```
assign_net_macros [define_region ....] net1 net2
```

net1

Net names are AFL level names. These names match your netlist names

most of the time. When they don't, you need to export AFL and use the AFL names. Net names are case insensitive. Hierarchical net names from ADL are not allowed.

Wild Cards are allowed for net names. The following regular expression syntax is supported:

- `\` interpret next character as non-special
- `?` matches any single character
- `*` matches any string
- `[]` matches any single character among those listed between brackets i.e. `[A-Z]` matches any single character in range A-Z.

Notes

1. The region must be created before you use the `assign_net_macros` command. See `define_region (rectangular)` or `define_region (rectlinear)`.
2. Placed macros (not spanned by the net) that are inside the area occupied by the net region are automatically unplaced.
3. Empty or exclusive regions cannot be created in areas where there are fixed macros.
4. Net region constraints are internally converted into constraints on macros. PDC export results as a series of `assign_region <region_name> macro1` statements for all the connected macros.
5. Net region constraints may result in a single macro being assigned to multiple regions. These net region constraints result in constraining the macro to the intersection of all the regions affected by the constraint.

Exceptions

- If the region does not have enough space, or if the region constraint is impossible, the constraint is rejected and a warning message appears in the log window.
- For overlapping regions, the intersection must be at least as big as the overlapping macro count.
- If a macro on the net cannot legally be placed in the region, it is not placed

and a warning message appears in the log window.

Example

```
assign_net_macros cluster_region1 keyin1intZ0Z_62
```

assign_region

The `assign_region` command constrains a set of macros to a specified region.

Syntax

```
assign_region  
    region macro1_name ?macron_name?...*
```

Arguments

region

Specifies the region where the macros are assigned. The macros are constrained to this region. The region must already be defined. Since the `define_region` returns a region object, you can write a simpler command such as:

```
assign_region ?define_region . . .? macro1 macro2 ...
```

macro1_name

Specifies the macro to assign to the region. Macro names are AFL level names. These names match your netlist names most of the time. When they do not, you need to export AFL and use the AFL names.

Macro names are case insensitive. Only Leaf level macros or HardMacros can be assigned (i.e. group names are not allowed).

Wild Cards are allowed for macro names. The following regular expression syntax is supported:

- `\` interpret next character as non-special
- `?` matches any single character
- `*` matches any string
- `[]` matches any single character among those listed between brackets i.e. `[A-Z]` matches any single character in range A-Z.

Note

- The region must be created before you can assign macros to it.

Exceptions

- If the region does not have enough space to fit the assigned macros, or if the region constraint is impossible to meet, the constraint is rejected and a warning message appears in the log window.

Example

```
assign_region cluster_region1 des01/G_2722_0_and2 des01/data1_53/U0
```

In this example, 2 macros are assigned to a region.

See Also

[unassign_macro_from_region_](#)

define_region (rectangular region)

The `define_region` command defines a rectangular region.

Syntax

```
define_region  
  -name ?region_name?  ?-type empty / exclusive / inclusive? x1 y1  
  x2 y2
```

Arguments

`?-name region_name?`

Specifies the name of the created region.. The region name is optional. If given, it must be unique. Do not use "bank0" and "bank<N>" for region names, as these are reserved. If a name is not specified, the system generates a name.

`?-type empty / exclusive / inclusive?`

Specifies the region type. The default is inclusive.

Region Constraint	Conditions
Empty	No macros can be assigned to an empty region
Exclusive	Only contains macros assigned to the region
Inclusive	Can contain macros both assigned and unassigned to the region

`x1, y1`

Specifies the lower left corner of the region in row, col

`x2, y2`

Specifies the upper right corner of the region in row, col

Notes

1. Empty and exclusive regions unplace macros in the area if they are not fixed.
2. Empty or exclusive regions cannot be created in areas where there are fixed macros.
3. Use exclusive regions carefully. They may over constrain the automatic place and route tool and reduce a design's performance.
4. Use inclusive or exclusive region constraints if you intend to assign logic to a region. An inclusive region constraint with no macros assigned to it has no effect.

An exclusive region constraint with no macros assigned to it is equivalent to an empty region.

5. Empty or exclusive regions cannot be created if a fixed macro exists within the area bounded by the newly defined region. A warning message appears in the log window.

Exceptions

- If macros assigned to a region exceed the area's capacity, an error message appears in the log window.
- If automatic place and route is unable to place logic within the area of the defined region an error message appears in the log window.
- Self-intersecting regions are not allowed. Rectangles must not intersect each other.

Example

```
define_region -name cluster_region1 -type empty 100 46 102 46
```

Defines a region with one rectangle and type empty.

See Also

`define_region` (rectilinear region)

define_region (rectilinear region)

The `define_region` command can also be used to define a rectilinear region or a rectangular region. This command applies only to the Axcelerator family.

Rectilinear regions are not supported for ProASIC and ProASIC PLUS.

Syntax

```
define_region  
  -name ?region_name? ?-type empty | exclusive | inclusive?  
  <rect1> <rect2> ...
```

Arguments

?-name *region_name*?

Specifies the region name. If the region cannot be created, the name is empty. The name must be unique. A default name is generated if a name is not specified in this argument.

?-type *empty | exclusive | inclusive*?

Specifies type. The default is inclusive.

Region Constraint	Conditions
Empty	No macros can be assigned to an empty region
Exclusive	Only contains macros assigned to the region
Inclusive	Can contain macros both assigned and unassigned to the region

<rect1> <rect2> ...

Specifies the series of coordinate pairs that constitute the region. These rectangles may or may not overlap. They are given as *x1 y1 x2 y2* (where *x1, y1* is the lower left and *x2 y2* is the upper right corner in row/column coordinates)

Notes

1. Empty and exclusive regions unplace macros in the area if they are not fixed.
2. Empty or exclusive regions cannot be created in areas where there are fixed macros.
3. Use exclusive regions carefully. They may over constrain the automatic place and route tool and reduce a design's performance.

4. Use inclusive or exclusive region constraints if you intend to assign logic to a region. An inclusive region constraint with no macros assigned to it has no effect. An exclusive region constraint with no macros assigned to it is equivalent to an empty region.
5. Empty or exclusive regions cannot be created if a fixed macro exists within the area bounded by the newly defined region. A warning message appears in the log window.

Exceptions

- If macros assigned to a region exceed the area's capacity, an error message appears in the log window.
- If place-and-route is unable to place logic within the area of the defined region, an error message appears in the log window.
- Self-intersecting regions are not allowed. Lines must not intersect each other.

Example

```
define_region -type empty 420 9 422 9  
  Defines a region with no name.
```

reset_floorplan

The `reset_floorplan` command deletes all created regions.

Syntax

```
reset_floorplan
```

Arguments

None

reset_io

The `reset_io` command restores all attributes on an I/O macro to the default values.

Syntax

```
reset_io  
    portname
```

Arguments

portname

Specifies the port name of the I/O macro to be reset.

Note: The *portname* can contain a limited CShell like regular expression using `?`, `*`, `[` and `]` characters.

Examples

```
reset_io a
```

Resets the I/O macro "a" to the default I/O attributes.

```
reset_io b_*
```

Resets all I/O macros beginning with `b_` to the default I/O attributes.

See Also

`set_io`

reset_iobank

The `reset_iobank` command resets an I/O bank's technology to the default technology, which is specified using the Designer software in the Device Selection Wizard.

Syntax

```
reset_iobank  
    bankname
```

Arguments

bankname

Specifies the I/O bank to be reset to the default technology. For the Axcelerator family, there are 8 banks numbered `bank0`-`bank7`.

Note

1. Any pins that are placed in the specified I/O bank that are incompatible with the default technology are unplaced.

Examples

```
reset_iobank bank4
```

Resets I/O bank 4 to the default technology.

See Also

`set_iobank`

reset_net_critical

The `reset_net_critical` command resets the net criticality to the default criticality. Net criticality can vary from 1 to 10, with 1 being the least critical and 10 being the most critical. Criticality numbers are used in timing driven place-and-route. The default net criticality is 5.

Increasing a net's criticality forces place-and-route to keep instances connected to the net as close as possible, at the cost of other (less critical) nets.

Syntax

```
reset_net_critical  
    netname ?netname? ...
```

Arguments

netname

Specifies the net name that is reset to the default net criticality.

Note

1. The *netname* can contain a limited CShell like regular expression using `?`, `*`, `[` and `]` characters.

Examples

```
reset_net_critical    preset_A  
    Resets the net preset_a to the default criticality of 5.
```

See Also

`set_net_critical`

set_io

The `set_io` command sets the attributes of an I/O. You can use the `set_io` command to assign an I/O technology, place, or fix the I/O at a given pin location.

Syntax

```
set_io
    portname ?-standard std? ?-slew high | low?
    ?-strength 8 | 12 | 16 | 24 ? ?-delay on | off?
    ?-register on | off? ?-pinname name?
    ?-fixed yes | no?
```

Arguments

portname

Specifies the portname of the I/O macro to set.

?-standard *std*?

Specifies the I/O standard to set for this I/O. Possible options include: LVTTTL, PCI, PCIX, LVCMOS25, LVCMOS18, LVCMOS 15, HSTL1, SSTL31, SSTL32, SSTL21, SSTL22, GLP33, GTLP25.

Note: Assigning an I/O standard to a port may invalidate its location. In this situation, the I/O will automatically be unplaced.

?-slew *high* | *low*?

Specifies the slew of this I/O.

?-strength *8* | *12* | *16* | *24*?

Specifies the output drive strength of this I/O.

?-delay *on* | *off*?

Specifies whether this I/O has an input delay.

?-register *on* | *off*?

Specifies whether register combining is allowed on this I/O.

Note: This overrides the default setting set in the Compile options.

?-pinname *name*?

Places this I/O macro in the specified pin.

?-fixed *yes* | *no*?

Sets whether the location of this I/O is fixed. Fixed pins are not moved during layout. If this I/O is not currently placed, then this argument has no effect.

Note

1. If an argument is not specified, the value is not changed, as long as it is consistent with other settings. If setting an attribute invalidates the I/Os location, then the I/O is unplaced.
2. The allowed I/O attributes vary from family to family. Please see the relevant device databook for details on allowed I/O attribute settings for a family.

set_iobank

The `set_iobank` command specifies the I/O bank's technology. Using this command may cause ports in the bank to be unplaced if they do not conform to the standard. The I/O bank name and all the parameters are required.

Syntax

```
set_iobank
    bankname ?-vcci vcci_voltage? ?-vccr vccr_voltage? ?-inputdelay
    delay_value?
    ?-lpininput On|Off? ?-lpoutput On|Off
```

Arguments

bankname
Specifies the bank. For the Axcelerator family, banks are numbered 0-7.

`-vcci vcci_voltage`
Sets voltage. Voltage should be set to 3.3, 2.5, 1.8.

`-vccr vccr_voltage`
Sets voltage. Voltage should be set to 1.5, 2.5,...

`-inputdelay delay_value`
Enables the input delay. Delay value can be set from 1-31.

`-lpininput On|Off`
Enables or disables the Low Power Mode for input buffers.

`-lpoutput On|Off`
Enables or disables the Low Power Mode for output buffers.

Examples

```
set_iobank bank0 -vcci 3.3 -vccr 1.5 -inputdelay 1 -
lpinput on -lpoutput on
```

```
bankname ?-vcci vcci_voltage? ?-vccr vccr_voltage? ?-inputdelay
delay_value? ?-lpininput On|Off? ?-lpoutput On|Off?
```

See Also

`reset_iobank`

set_location

The `set_location` command places a specified logic instance at a particular location.

Syntax

```
set_location  
  <clustername> -fixed yes x y
```

Arguments

clustername x y

Specifies the module instance name in the netlist. *x* and *y* set the module instance name coordinates. The coordinates *row/col* are the same as seen in ChipEditor.

Note

- The instance names are the post-compiled names after name translations, if any.

set_net_critical

The `set_net_critical` command sets the net criticality, which influences place-and-route in favor of performance.

Syntax

```
set_net_critical
```

```
criticality_number hier_net_name ?hier_net_name?
```

Arguments

criticality_number

Sets the criticality level. Set from 1 to 10. Default is 5. Net criticality can vary from 1 to 10, with 1 being the least critical and 10 being the most critical. Criticality numbers are used in Timing Driven place-and-route. The default net criticality is 5.

Increasing a net's criticality forces place-and-route to keep instances connected to the specified net as close as possible at the cost of other (less critical) nets.

hier_net_name

Specifies the net name. Can be an AFL net name or a net regular expression (a limited CShell like regular expression using `?`, `*`, `[` and `]` characters).

Notes

1. The command must have at least 2 parameters.
2. The net names are AFL names, meaning they must be visible in Timer and ChipEditor.

Examples

```
set_net_critical 9 addr*
```

Sets the criticality level to 9 for all `addr` nets.

See Also

`reset_net_critical`

set_vref

The `set_vref` command specifies which pins are Vref pins.

Syntax

```
set_vref  
    ?bankname? pinnum ...
```

Arguments

?bankname?

Specifies the bank name. For the Axcelerator family, banks names are 0-7 (Bank0, Bank1, ... Bank7).

pinnum

Specifies the alphanumeric pin name.

Notes

1. While the bank name is optional, you must not mention pin names that do not belong to the bank. Pins that do not belong to a bank that require a Vref are ignored.
2. Some I/O technologies need Vref settings. Some technologies may also need a minimum number of Vref pins for every certain number of input pins. These details are device dependent. Please refer to the device databook for details. Designer can assign default Vref pins. However, this may be too conservative, and you may not need as many Vref pins as the default assignment.

Examples

```
set_vref A1 B10
```

See Also

`set_vref_default`

set_vref_defaults

The `set_vref_defaults` command sets the default vref pins for the specified bank. This command is ignored if the bank does not need a Vref.

Some I/O technologies need Vref settings. Some technologies may also need a minimum number of Vref pins for every specific number of input pins. These details are device dependent. Please refer to the device databook for details. The Designer software can assign default Vref pins. However, this may be too conservative, and you may not need as many Vref pins as the default assignment.

Syntax

```
set_vref_defaults  
    bankname
```

Arguments

bankname

Specifies the bank name. For the Axcelerator family, banks names are 0-7 (Bank0, Bank1, ... Bank7).

Examples

```
set_vref_defaults bank1
```

Sets the default Vref pins for Bank 1.

See Also

`set_vref`

unassign_net_macro

The `unassign_net_macros` command unassigns macros connected to a specified net.

Syntax

```
Unassign_net_macros  
    region_name net1 ?netN?
```

Arguments

region_name

Specifies the region name.

net1

Specifies the net name.

?netN?...

Optional argument, specifies additional net macros to be unassigned.

Note

1. If you have not assigned the region, an error message appears in the log window.

See Also

`assign_net_macros`

unassign_macro_from_region

The `unassign_macro_from_region` command specifies the macro name to be unassigned.

Syntax

```
unassign_macro_from_region  
    ?region_name? macro_name
```

Arguments

?region_name?

Specifies the region where the macro or macros are to be removed.

macro_name

Specifies the macro name to be unassigned. Macro names are case sensitive. Only Leaf level macros or HardMacros can be assigned (i.e. group names are not allowed).

Wildcards are allowed for macro names. Hierarchical net names from ADL are not allowed.

The following regular expression syntax is supported

- `\` interpret next character as non-special
- `?` matches any single character
- `*` matches any string
- `[]` matches any single character among those listed between brackets i.e. `[A-Z]` matches any single character in range A-Z.

Note

1. If the macro was not previously assigned, an error message is generated.

undefine_region

The `undefine_region` command removes the specified region.

Syntax

```
undefine_region  
    region_name
```

Arguments

region_name
Specifies the region to be removed.

Notes

1. The region must be previously defined.
2. All macros assigned to the region are unassigned.

Example

```
undefine_region cluster_region1
```

See Also

`define_region` (rectangular region)
`define_region` (rectilinear region)

PDC Errors

After executing a PDC Tcl command, a message or error might appear in the Designer log window. Below is a list of possible errors and workarounds.

“ERROR: Error in setting I/O Standard”

This error may happen because the device does not support the I/O standard.

“ERROR: I/O Attribute is not applicable”

You specified an incorrect or invalid I/O attribute name in the `set_io` command. Please check the spelling. Not all devices support the same set of I/O attributes.

“ERROR: Illegal or Invalid assignment to Package pin”

The specified assignment may be illegal because the bank associated may not be able to support the given technology or the specified location does not match the type of the given port.

“ERROR: Unknown port”

You specified an incorrect portname in the `set_io` command. Check the spelling. Also, make sure to escape any Tcl special characters (TCL Special characters are \$, “, [,], {, }, \ - you can escape by \)

“ERROR: Unknown I/O Standard”

You specified an incorrect I/O standard. This device may not support the standard you specified or you have spelled it incorrectly.

“ERROR: Read only I/O Standard”

Designer can not change the I/O standard (and other properties) of a port that has been set either in the library or schematic or HDL. You need to change the I/O standard in the Library/Schematic/HDL and re-import the netlist.

“ERROR: Unknown I/O Attribute”

An incorrect or invalid I/O attribute name was specified when using the `set_io` command, Please check the spelling. Note that not all devices support the same set of I/O attributes.

“ERROR: Read only I/O Attribute”

The Designer software can not change the I/O standard (and other properties) of a port that has been set either in the library or schematic or HDL. You need to change the I/O standard in the Library/Schematic/HDL and re-import the netlist

“ERROR: Invalid Package pin”

This error is caused when an illegal or incorrectly spelled package pin name is used with the `set_io` command. Check the spelling and make sure to escape Tcl characters, if any.

“ERROR: Net criticality must be 1-10”

Net criticality must be set between 1 and 10.

“Error: Unknown Net”

This error occurs when an unknown net is specified in the `set_net_critical` or `reset_net_critical` commands. Check your spelling and make sure to escape Tcl special characters.

“Warning: Some ports have been unplaced because of this action”

The technology you assigned to the bank is incompatible with the ports already in the bank. As a result, the ports were unplaced.

Timing Constraints

Timing constraints can be entered using the interactive Timer tool or by importing a constraint file.

Constraint File Type	Supported Families
SDC	Axcelerator
DCF	SX, SX-A, MX, eX, ACT1, ACT2, and ACT3
GCF	Flash

Synopsys Design Constraints (SDC) Files

SDC is a Tcl based format-constraining file. The commands of an SDC file follow the Tcl syntax rules. Designer accepts an SDC constraint file generated by a third-party tool. This file is used to communicate design intent between tools and provide clock and delay constraints. The Synopsys Design Compiler, Prime Time, and Synplicity tools can generate SDC descriptions, or you can generate the SDC file manually.

Generated SDC files

There can be slight differences between a user generated SDC file, and SDC files generated by other tools. For example, suppose you write the following constraint:

```
create_clock -period 100 clk
```

The SDC file from Design Compiler generates the same constraint in a different format:

```
create_clock -period 100 -waveform {0 50}
[get_ports {clk}]
```

The SDC file from Prime Time generates this constraint in yet another format:

```
create_clock -period 100.000000 -waveform
{0.000000\ 50.000000}[get_ports {clk}]
```

As long as constraint syntax and arguments conform to the Tcl syntax rules that SDC follows, Designer will accept the SDC file.

Importing Constraint (Auxiliary) Files

The following constraint file types can be imported into Designer:

Auxiliary Files	File Type Extension	Family
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC ^{PLUS}
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC ^{PLUS}
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

To import an auxiliary file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog appears,
2. Click the **Add** button. The Add Auxiliary Files dialog box appears.
3. Select your file and click **Import**. The file is added to the Import Auxiliary Files dialog box. Continue to add more auxiliary files to the list.
 - **Modifying:** If you need to modify a selection, select the file row and click

Modify

- **Deleting:**If you need to delete a file, select the file row and click Delete.
 - **Ordering:**Ordering your source files. Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files.
4. After you are done adding all your Auxiliary files, click **OK**. Your auxiliary files are imported. Any errors appear in Designer's Log Window.

Note:

1. .vcd and .saif are used by SmartPower for power analysis.
2. .crt for backwards compatibility with existing designs only.
5. File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.

SDC Commands

Designer supports some some SDC commands.

Design Object Access Commands

Design object access commands are SDC commands. Most constraint commands require a command argument. Designer supports the SDC access commands shown below:

Design Object	Access Command
Clock	get_clocks
Port	get_ports

get_clocks

Returns the named clock with the argument.

Example:

```
create_clock -period 10 [get_clocks CK1]
```

get_ports

Returns the named ports with the argument

Example:

```
set_max_delay -from [get_ports data1] -to  
[get_ports out1]
```

Timing constraint commands

Designer supports the SDC timing constraint commands below:

Constraint	Command
Clock Constraint	create_clock
Path Constraint	set_max_delay

create_clock

The create_clock constraint is associated with a specific clock in a sequential design and determines the maximum register-to-register delay in the design.

Syntax

```
create_clock -period period_value [-name  
clock_name] [-waveform edge_list]  
[port_pin_list]
```

Arguments

period_value

Specified in ns is mandatory. No clock is created if the period is not supplied.

`clock_name`

This is optional. It is unnecessary if `port_pin_list` contains one name.

`edge_list`

This is optional and not supported in the current version of Designer. If supplied, it must contain exactly 2 edges. The duty cycle info will then be added to the clock constraint.

`port_pin_list`

May contain either zero names or one name.

Valid Command Examples

```
create_clock -period 5 -name CK1
```

```
create_clock -period 4 -name CK1 -waveform 0 2
```

```
create_clock -period 6 [get_ports CK1]
```

```
create_clock -period 11 -name CK1 -waveform 0 2 5 7
```

This command is valid, but the waveform will be ignored.

```
create_clock -period 2 -name CLOCK [get_ports CK1]
```

This is valid, but the name of the clock will be CK1 and not

CLOCK

Invalid Command Examples

```
create_clock -period 10 (no name is supplied)
```

```
create_clock -period 3 [get_ports {CK1 CK2}]
```

This command is invalid because more than one name is used in the `port_pin_list`

```
create_clock -period 7 -name CK [get_ports {CLK11  
CLK2}]
```

```
create_clock clk -name CLK1 -period 20 -waveform 3 13
```

set_max_delay

The set_max_delay constraint sets the path delay of the specified ports to a restricted value.

Syntax

```
set_max_delay [-from from_list] [-to to_list]  
delay_value
```

Arguments

```
from_list  
    Mandatory  
to_list  
    Mandatory  
delay_value  
    Specify in nano seconds
```

Valid Command Examples:

```
set_max_delay -from [get_ports data2] -to [get_ports {out1 out2}] 9
```

Invalid Command Examples

```
set_max_delay -from [get_ports {IN10 IN11}]5 (The to_list is not supplied.)
```

SDC Command limitations

Not all object and design constraint commands are supported in Designer. There are limitations on SDC support. Refer to the latest Designer series Release Notes for latest supported Object Access, Design Constraints, and Supported Features.

Naming Conventions

No wild cards. The * and ? characters cannot be used in the object names. The timing graphical interface, Timer, displays internal Actel port names. While the internal Actel netlist prevents special characters from being used, in the case where the internal name is different from the “user” netlist, there may be discrepancies in the GUI. These could also be different from the names in the SDC files.

Multiple Files

All the constraints have to be imported from a single SDC file. If a second file is imported, the previous constraints are discarded.

GCF Files

GCF files can import timing constraints information. Timing constraints are used to ensure that a design meets the required timing performance. Constraints can be entered using a ProASIC constraints file (.gcf) or using an SDF path constraints file. These forward SDF files are generated by synthesis tools. The two formats cannot be combined in one file. However, SDF files and ProASIC (.gcf) constraint files can be used for the same design. Place and Route considers timing constraints and attempts to meet them.

After routing, Designer displays messages to identify the constraints that cannot be met.

Importing GCF files

Import GCF files as you would any source file.

To import a GCF file:

1. In the **File** menu, click **Import Source Files**.
2. Click **Add**. The Add Source Files dialog appears.
3. Select ProASIC Constraint Files (.gcf) from Files of type.
4. Select your .gcf file and click **Import**. The File is added to the Import Source Files dialog box.
5. Add any more source files to the list. All files added to the Import Source Files dialog box are imported at the same time.
 - **Modifying:** If you need to modify a selection, select the file row and click **Modify**
 - **Deleting:** If you need to delete a file, select the file row and click **Delete**.

- **Ordering:** Ordering your source files. Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files. When importing multiple EDIF or structural HDL files, the top-level file must appear last in the list (at the bottom).
6. After you are done adding all your source files, click **OK**. Your source files are imported. Any errors appear in the Designer Log Window.

Timing Constraints Guidelines

To understand the complexity of a design and its performance, perform placement and routing with no constraints to see if routing can complete without constraints. If routing completes successfully, create the timing annotation files and back-annotate the post-layout delays to see if the physical design meets timing requirements. If you are using a synthesis tool such as Synopsys Design Compiler, Actel recommends that you use it to generate a forward SDF file containing path constraints only.

If these requirements are not met, you can guide timing driven place and route by forward annotating the SDF generated by the synthesis tool. Timing constraints must be reasonable. Over constraining a design may result in increased place and route run times, while not improving circuit performance.

Highlevel Timing Constraints

create_clock

Use this statement to define clocks for the design. Multiple clocks can be specified for a given design.

```
create_clock -period <period_value> {netname|portname}
```

Where “period_value” is the clock period in nanoseconds and “netname|portname” is the name of the net through which the clocks gets propagated or name of the external port.

For example, the following statement creates a clock on external port “clk” with a period of 25.0 nanoseconds.

```
create_clock -period 25.0 clk;
```

generate_paths

Use this statement to modify the way Designer generates internal path constraints for the placer to do timing driven placement.

```
generate_paths [-cover_design] [-max_paths <maxpaths>
[-top <percentage>];
```

Where “-cover_design” indicates to Designer to use the “cover design” algorithm instead of the default worst paths algorithm, “-max_paths” is the maximum number of paths that will be generated (default is 20% of the number of nets with minimum of 1000 or if cover_design is specified twice the number of nets with a minimum of 1000), “-top” indicates the top percentage of worst paths that will be generated (default is 20%).

For example, the following statement generates 4000 maximum paths using the -cover_design algorithm.

```
generate_paths -cover_design -max_paths 4000;
```

set_false_path

Use this statement to define false paths in the design. These paths are not considered in the timing driven place and route system.

```
set_false_path [-from from_port] [-through any_port] [-to to_port];
```

Where “from_port” must be an input port of the design or a register or memory instance output pin, “to_port” must be an output port of the design or a register or memory instance input pin, “any_port” must be any instance pin. Wildcards are permitted.

For example, the following statement sets all paths starting from “resetd” which are going through instance “const2” as false paths.

```
set_false_path -from resetd -through const2/*;
```

set_input_to_register_delay

Use this statement to define the timing budget for incoming signals to reach a register:

```
set_input_to_register_delay <delay> [-from inp_port];
```

Where “delay” is the timing budget for this input path, “inp_port” is a register or memory instance output pin. Wildcards are permitted.

For example, the following statement specifies that the timing budget is 22 nanoseconds to the register from all inputs whose names are starting with letter “I”.

```
set_input_to_register_delay 22 -from I*;
```

set_multicycle_path

Use this constraint to define how many clock cycles a signal has to travel through these paths. The budget of these paths will be a multiple of the period of the clock controlling the from port.

```
set_multicycle_path <num_cycles> -from reg_port [-through_any_port] [-to_port];
```

Where “num_cycles” is the number of clock cycles in which the signal needs to propagate through the path, “reg_port” is a register or memory instance, “to_port” must be an output port of the design or a register or memory instance input pin, “any_port” must be any instance pin. Wildcards are permitted.

For example, the following statement specifies it takes two clock cycles to reach signals from instance pins /us/u1/dff*.q to instance pins /u4/mem1/*.D.

```
set_multicycle_path 2 -from /us/u1/dff*.q -to /u4/mem1/*.D;
```

set_register_to_output_delay

Use this statement to define the timing budget for outgoing signals to be clocked out.

```
set_resgister_to_output_delay <delay> -to out_port;
```

Where “delay” is the timing budget for this output path, “out_port” must be an output port of the design. Wildcards are permitted.

For example, the following statement specifies the timing budget for clocking out signals on output ports starting with “O” is 22 nanoseconds.

```
set_register_to_output_delay 22 -to O*;
```

Timing Constraints

net_critical_ports

Use this statement to specify a specific subset of critical ports on a net.

For example, the following statement identifies two inputs of the net “/u1/u2/net1” that are more critical than all other connections on that net. All other connections on the net will be buffered with a “BUF” cell that will be placed in a tile to reduce fanout delay on the specified inputs:

```
net_critical_ports /u1/u2/net1 nandbk1.A sigproc.C;
```

set_critical

Use this statement to specify critical nets and their relative criticality over other critical nets.

```
set_critical criticality_number hier_net_name  
[,hier_net_name ...];
```

Where “criticality_number” is from 1 to 5 (1 being the default criticality for every net and 5 the highest). “hier_net_name” is the full hierarchical net name.

For example, the statements below set the timing of “u1/u2/ net1” more critical than “u1/u2/net5 and u1/u2/net3”:

```
set_critical 5 /u1/u2/net1;  
set_critical 2 /u1/u2/net5, u1/u2/net3;
```

set_critical_port

Use this statement to identify design I/O ports that have above-normal criticality. The criticality number scales is the same for the “set_critical” statement.

```
set_critical_port criticality_number signal_name  
[,signal_name ...];
```

Where “signal_name” is the name of a user-defined signal associated with a specific I/O pin on the part.

For example, the following statement sets all nets associated with device ports IOBus[3] and IOBus[5] to have criticality 3:

```
set_critical_port 3 IOBus[3], IOBus[5];
```

set_max_path_delay

Use this statement to constrain the maximum delay on paths. The calculate timing task will report a note in the timing report file if this delay is not met.

```
set_max_path_delay delay_value  
hier_inst_name_.inst_port_name  
[, hier_inst_name .inst_port_name , ... ];
```

Where “delay_value” is a floating integer for delay in nanoseconds, “hier_inst_name” is the hierarchical path to a cell instance, and “inst_port_name” is a port name of a cell instance.

For example:

```
set_max_path_delay 12.5 "mult4/mult/nand2_2".Y, "mult4/mult/  
nand3_1".A, "mult4/mult/nand3_1".Y, "mult4/mult/nor2_2".A;
```

set_switch_threshold

Use this statement to specify the number of switches the router is allowed to route a net through, before it has to insert an active repeater while routing the specified net. The default for all nets is 8.

```
set_switch_threshold <threshold> <net_name>;
```

Where “threshold” is a integer for the threshold, range 4 to 16, and “net_name” is the name of the net(s) the threshold should be used for. Wildcards are permitted.

For example:

```
set_switch_threshold 6 core/fsm/state_1;  
set_switch_threshold 6 core/fsm/state_*;
```

Design Implementation

The Designer interface offers both automated and manual flows, with the push-button flow achieving the optimal solution in the shortest cycle.

Actel's Designer software is integrated with Libero IDE. Use the Designer software to implement your design.

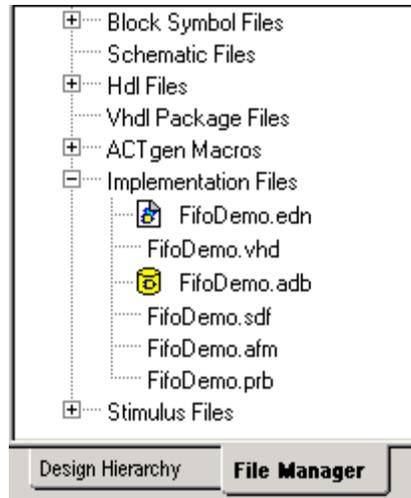
To implement your design:

1. **Start Designer.** Right-click the top level module in the **Design Hierarchy** and select **Run Designer**, or double click **Designer** in the Process window. Designer starts and loads your files from Libero.
2. Set up your device. From the Tools menu, click Device Selection. In the Device Selection Wizard, select your device type, device package, speed grade, voltage, and operating conditions. Make your selections and click Next to complete the steps
3. **Compile your design.** In Designer, click **Compile** in the design flow window. The log window displays the utilization of the selected device. When compile has completed, the Compile box in the Design Flow window turns green.
4. **Designer's User Tools.** Once you have successfully compiled your design, you can use Designer's User's tool to optimize your design. To start a tool, simply click it in the flow tree. The tools include:

Tool	Function	Supported Families
PinEditor	Package level floorplanner and I/O attribute editor	All

ChipPlanner	Logic viewer, placement and floorplanning tool	Axcelerator,Flash
ChipEditor	Logic viewer and placement tool	All
NetlistViewer	Design schematic viewer	All
SmartPower	Power analysis tool	Axcelerator, Flash
Timer	Static timing analysis and constraints editor	All

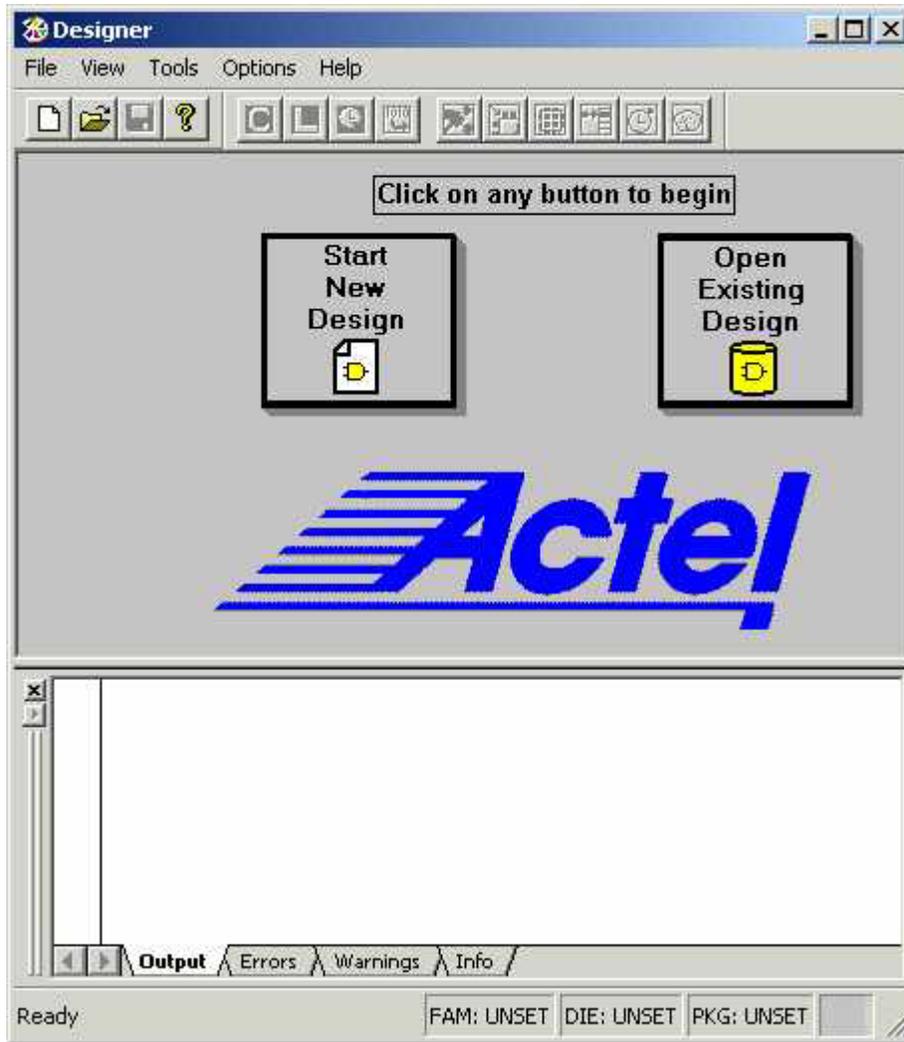
5. **Layout your design.** Click **Layout** in the Design Flow Window to place-and-route your design.
6. **Back-Annotate your design.** Click **Back-Annotate** in the Design Flow Window. Choose SDF as CAE type and appropriate simulation language. Select Netlist in the Export Additional Files area and Click **OK**. If you are exporting files post-layout, Designer exports <top>_ba.vhd and <top>_ba.sdf to your Libero project. The “_ba” is added by Libero to identify these for back-annotation purposes. <top> is the top root name. Pre-layout exported files do not contain “_ba” and are exported simply as *.vhd and *.sdf. The files are visible from the File Manager, under Implementation Files.
7. **Generate a programming file.** Click **Fuse** or **Bitstream** in the design flow tree if you wish to create a programming file for your design. This step can be performed later after you are satisfied with the back-annotated timing simulation.
8. **Save and Exit.** From the **File** menu, click **Exit**. Select Yes to save the design before closing Designer. Designer saves all of the design information in an *.adb file. The <project>.adb file is visible in Libero’s File Manger, in the Implementation Files folder. To re-open this file at any time, simply double-click it.



Starting Designer

To start Designer from Libero IDE:

In the process window, click Designer Place-and-Route.



Designer

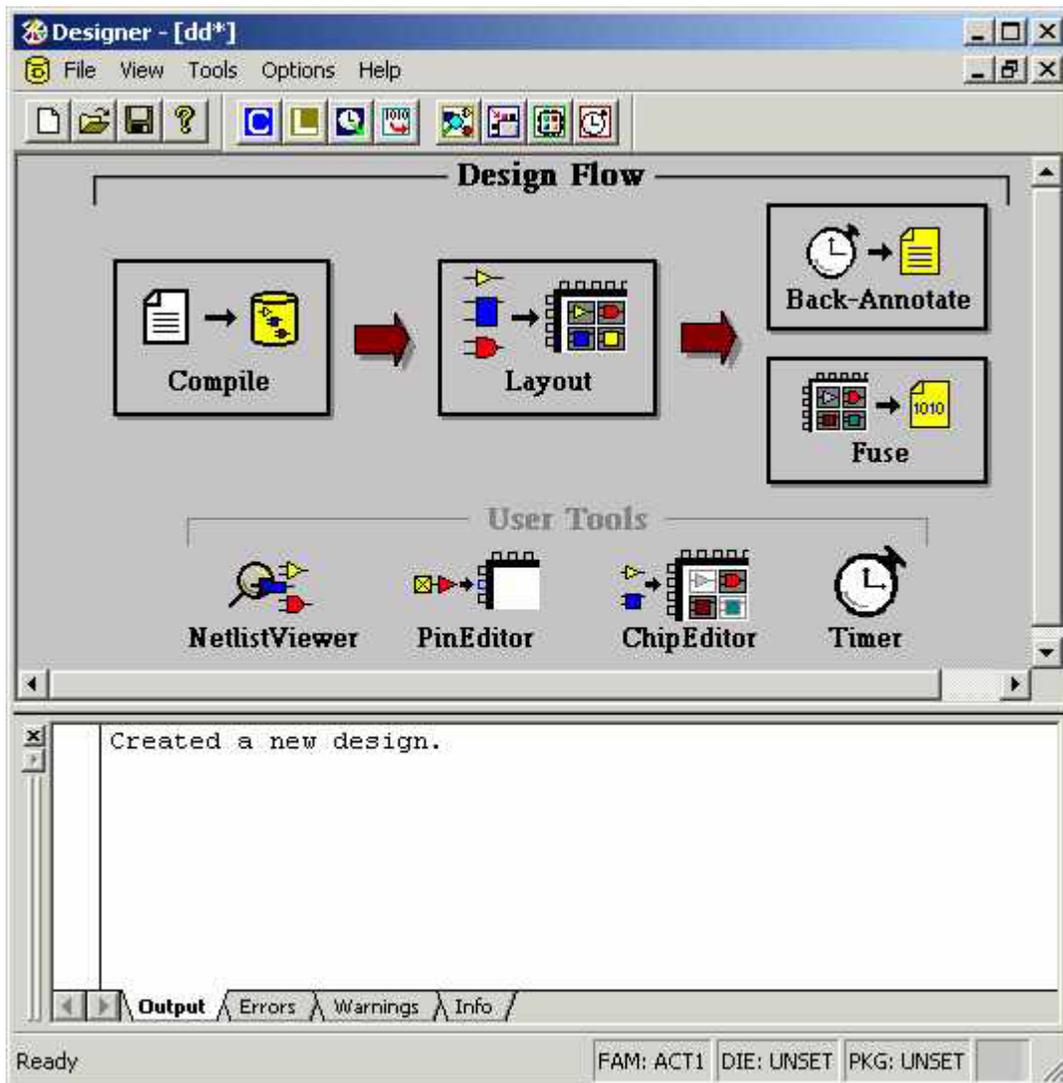
Starting a new design

To begin a new design session, you must start a new design or open an existing design.

To start a new design:

1. Click **Start New Design** in the Designer main window, or in the **File** menu, click **New**. This displays the Setup Design dialog box.
2. Setup Design:

- Enter a **Design Name**. The design name is used in reports and as the default name when saving or exporting files.
 - Select an **Actel Family** from the drop down menu list.
 - Specify a **working directory**. Click Browse to locate a directory.
3. Click **OK**. The Designer custom design flow window appears. All tools and commands are activated.



Designer:New Design

Opening an existing design

To open an existing design:

1. Click **Open Existing Design** or in the **File** menu, click **Open**. This displays the Open dialog box
2. Select **File**. Type the full path name of the .adb file in the File Name box, or select the file from the list.
3. Click **Open**. Designer's custom design flow window appears and all tools and commands are activated. When you open an existing design, Designer checks to see if you have modified your netlist since the last time you imported the netlist into this design. If you have, Designer prompts you to re-import your netlist.

Opening designs created in previous versions

Designer can directly open designs created with previous versions of the Designer software.

If your design was created in version 3.1 or earlier, contact Applications or go to <http://www.actel.com/support> for information on converting your design.

All existing die, package, pin assignments, and place-and-route information is read and maintained. Designs created in previous versions of software may need library conversions when loaded into the Designer environment. If your design requires this conversion, Designer prompts you to allow the software to update the design to the new library before you attempt to start any of the Designer features.

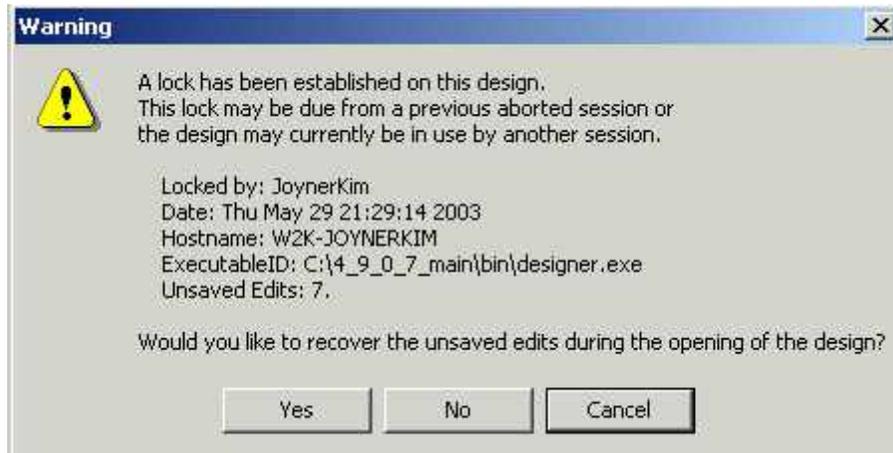
Opening locked files

Designer notifies you if a lock has been established on your file. You might get a warning or an error message when opening a design with a lock.

Warning

Designer warns you when opening a design that was not closed properly or may be open

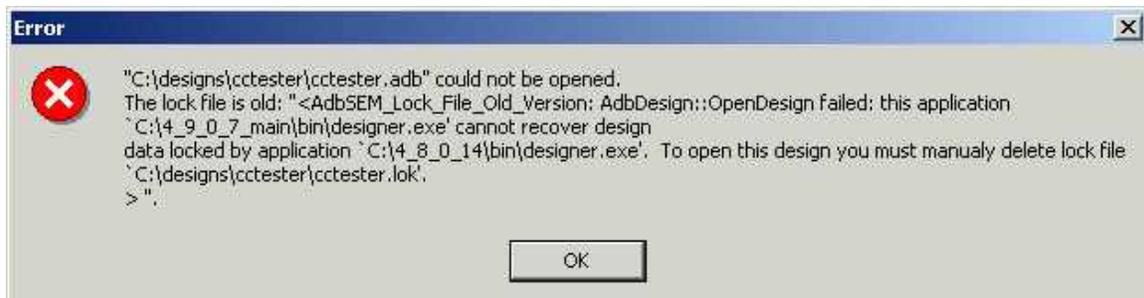
somewhere else. You can choose to recover the unsaved edits.



Warning: Locked File

Error

When opening a design, an error might notify you that the file can't be opened because the lock file is old. You can't recover any unsaved edits.



Error: Locked File

To open a design with an old lock file:

1. Go to the design directory.
2. Locate the design .adb file and corresponding .lok file.
3. Delete the .lok file.
4. Return to Designer and open the design.

Name	Size	Type	Modified
ada01928.4	87 KB	4 File	2/7/2003 10:24 PM
cctester.adb	87 KB	Actel Designer Desi...	12/11/2001 10:39 AM
cctester.lok	1 KB	LOK File	2/7/2003 10:24 PM

Starting other applications from Designer (PC only)

You can start any application from Designer that you have added to the Tools menu.

To add an application to the Tools menu:

1. From the **Tools** menu, click **Customize**.
2. Enter the application name in the Menu Text area. This text will appear in the Tools command menu.
3. Enter the command to execute, or click the **Browse** button to select an executable filename. If the location of the command to execute is not in your path, you must include the absolute path when specifying the command.
4. In the Arguments text box, enter the command-line arguments that will be passed to the command when executing.
5. In the Initial Directory field, type the absolute path of the directory in which the application will initially be executed.
6. Click **Add**.
7. When you are finished adding tools, click **OK**. The application name you added appears in the Tools menu.

To remove an application from the Tools menu:

1. From the **Tools** menu, click **Customize**.
2. Select the application to remove and click **Remove**.

3. When you are finished removing applications, click **OK**.

To order applications in the Tools menu:

1. From the **Tools** menu, click **Customize**.
2. Reorder the tools by selecting one at a time and clicking the **Move Up** or **Move Down** buttons.
3. Click **OK** when you are finished. The tools will appear in the Tools menu in the same order as they do in the Menu Contents list box.

License details

To display information about your license:

1. Open your project or start a new one.
2. From the help menu, click **License Details**. The License Details dialog box appears. This information cannot be edited, it is for display purposes only.

Preferences

Directory preferences

When executing a command or function such as Open or Save, Designer uses the directory you specify as the start-up directory.

To specify your directory preferences:

1. From the **File** menu, click **Preferences**.
2. Click the **Directory** tab.
3. Specify your Startup directory.
4. Select your working directory options:
 - **To design file's directory when opening design:** Select to automatically change directories when opening a design.

- **To design file's directory when saving design:** Select to automatically change directories when saving a design.
- **To script file's directory when executing script:** Select to automatically change directories when executing a script.
- **Add design name to working directory when creating design:** Select to enable a design name folder to be automatically created in the working directory when creating a new design.

5. Click OK.

Updates

The Updates tab in the Preferences dialog box allows you to set your automatic software update preferences.

To set your automatic software update preferences:

1. From the **File** menu, click **Preferences** and **Updates**.
2. Choose one of the following options and click **OK**.
 - **Automatically check for updates at startup:** Select to be notified of updates when you start Designer.
 - **Remind me to check for updates at startup:** Select to be asked if you want to check for a software update when you start Designer.
 - **Do not check for updates or remind me at startup:** Select if you do not want to check for software updates at startup.

To manually check for software updates, from the **Help** menu, click **Check for Software Updates**.

Note:

- This feature requires an internet connection.

Proxy

A Proxy improves access to the Actel server.

To enable the proxy:

1. Select I use a proxy.
2. Type the proxy name in the text field.
3. Click OK.

File association

Several programs, including Designer, create files with the .adb extension.

Use the File Association tab in the Preferences dialog box to specify Designer as the default program for files with the .adb extension. Doing so starts Designer whenever a file with the .adb extension is double clicked.

To associated .adb files with the Designer application:

1. From the **File** menu, click **Preferences**.
2. Select Check the default file association (.adb) at startup to Check the box to associate .adb files with the Designer application. Un-check the box if you do not want Designer to start when clicking a file with the .adb extension.
3. Click OK.

Setting your Log Window preferences

Errors, Warnings, and Informational messages are color coded in the log window. You can change the default colors by using the log Window tab in the Preferences dialog box.

To change colors in the log window:

1. From the **File** menu, choose **Preferences**.

2. Click the **Log Window** tab in the Preferences Dialog Box.
3. Select your new default colors and click **OK**.

The default color settings for the log window are:

Message Type	Colors
Errors	Red
Warnings	Light Blue
Informational	Black
Linked	Dark Blue

PDF Reader (UNIX Only)

Use the PDF Reader tab to bring up the Designer online manuals. Enter the default reader's name with the full path or click browse.

Device Selection

Device Selection Wizard

After you import your source files, the Device Selection Wizard helps you specify the device, package, and other operating conditions. You must complete these steps before your netlist can be compiled.

The wizard steps include:

- Selecting die, package, speed, and voltage
- Selecting variations (reserve pins and I/O attributes)

- Setting operating conditions

Setting die, package, speed, and voltage

The first screen in the Device Selection Wizard allows you to set die, package, speed, and voltage.

1. In the **Tools** menu, click **Device Selection** to start the Device Selection Wizard.
2. Select **die** and **package**. Select a die from the Die list. Available packages are listed for each die.
3. Specify **speed**.
4. Select **die voltage**. Select from the available settings in Die Voltage drop-down menu. Two numbers separated by a “/” are shown if mixed voltages are supported. If two voltages are shown, the first number is the I/O voltage and the second number is the core (array) voltage
5. Click **Next** to set reserve pins and I/O Attributes.

Device variations

The second screen in the Device Selection Wizard enables you to set reserve JTAG and probe pins and the default I/O standard.

To select reserve pins and default I/O standard:

1. Select your reserve pins:
 - Check the Reserve JTAG box to reserve the JTAG pins “TDI,” “TMS,” “TCK,” and “TDO” during layout.
 - Check the Reserve JTAG Reset box to reserve the JTAG reset Pin “TRST” during layout.
 - Check the Reserve Probe box to reserve the Probe pins “PRA,” “PRB,” “SDI,” and “DCLK” during layout.

Reserve Pins are not selectable for the Axcelerator, ProASIC, and ProASIC ^{PLUS} families.

2. Select an I/O attribute. The I/O Attributes section notifies you if your device supports the programming of I/O attributes on a per-pin basis. For the Axcelerator family, the I/O Attribute section allows you to set the default I/O standard for the I/O banks.
3. Click **Next** to set operating conditions.

Setting Operating Conditions

Operating Conditions, step 3 of the Device Selection Wizard, enables you to define the voltage and temperature ranges a device encounters in a working system. The operating condition range entered here is used by Timer, the timing report, and the back-annotation function. These tools enable you to analyze worst, typical, and best case timing.

Junction Temperature

Select a junction temperature. Supported ranges include:

- Commercial (COM)
- Industrial (IND)
- Military (MIL)
- Automotive
- Custom

Consult the Actel Data Sheet, available at <http://www.actel.com/techdocs/ds/index.html> to find out which temperature range you should use.

If you select Custom, edit the Best, Typical, and Worst fields. Modify the range to the desired value (real) such that Best < Typical < Worst.

Voltage

Select a voltage:

- Commercial (COM)
- Industrial (IND)
- Military (MIL)
- Automotive
- Custom

You can calculate junction temperature from values in the Actel Data Sheet, available at <http://www.actel.com/techdocs/ds/index.html>.

The temperature range represents the junction temperature of the device. For commercial and industrial devices, the junction temperature is a function of ambient temperature, air flow, and power consumption.

For military devices, the junction temperature is a function of the case temperature, air flow, and power consumption. Because Actel devices are CMOS, power consumption must be calculated for each design. For most low power applications (e.g. 250mW), the default conditions should be adequate.

Performance decreases approximately 2.5% for every 10 degrees C that the temperature values increase. Refer to the SmartPower User's Guide for more information about power consumption.

Radiation Derating

Conservative post radiation performance estimates are available for some radiation tolerant devices based upon the number of KRads the device is expected to be subjected to. Radiation effects vary by device lot and may not be completely representative of the lot you are using. Post radiation timing numbers are only meant to be a guide and are not a guarantee of performance. Customers must consult the specific radiation performance report for the specific lot used. Post radiation exposure estimates currently only affect timing numbers. The SmartPower power analysis tool is not affected by changing the radiation exposure value.

Changing Design name and family

Design name and family are set when you import a netlist and compile a new design. However, you can change this information for existing designs. If you change the family, Designer notifies you that you must re-import the netlist and automatically prompts you when you select the next Designer function. Use the following procedure to change the name of a design and the targeted Actel family for the design.

To change the design name or family:

1. In the **Tools** menu, click **Setup Design**. This displays the Setup Design dialog box.
2. Specify the design name and family.
3. Click **OK**. Refer to the Actel FPGA Data Book for Actel Family specifications.

Changing device information

Device and package information, device variations, and operating conditions are set when you import a netlist and compile a new design. However, you can change this information for existing designs.

To change design information for existing designs:

1. In the **Tools** menu, click **Device Selection**. The Device Selection Wizard appears.

2. Select Die, Package, and Speed Grade and click **Next**. (You must select die and package to continue.)
3. Select Device Variations and click **Next**.
4. Select Operating Conditions and click **Finish**.

Refer to the Actel FPGA Data Book or call your local Actel Sales Representative for information about device, package, speed grade, variations, and operating conditions.

Changing Device, Package, and Speed Grade

Use the Device Selection dialog box to specify or change the device and package type and the speed grade based on your design needs. Refer to the Actel website for the latest information (<http://www.actel.com>). If you select a device, available packages are then displayed in the Package list box. If you select a package, specify a speed grade in the Speed Grade pull-down menu.

Devices that are no longer available from the Device Selection dialog box can be selected using Designer Script. Because these parts may no longer be available, do not use these devices unless approved by Actel.

Compatible Die Change

When you change the device, some design information can be preserved depending on the type of change.

Changing Die Revisions

If you change the die from one technology to another, all information except timing is preserved. An example is changing an A1020A (1.2um) to an A1020B (1.0um) while keeping the package the same.

Device Change Only

Constraint and pin information is preserved, when possible. An example is changing an A1240A in a PL84 package to an A1280A in a PL84 package.

Repackager Function (Non-Axcelerator families only)

When the package is changed (for the same device), the Repackager automatically

attempts to preserve the existing pin and Layout information by mapping external pin names based on the physical bonding diagrams. This always works when changing from a smaller package to a larger package (or one of the same size). When changing to a smaller package, the Repackager determines if any of the currently assigned I/Os are mapped differently on the smaller package. If any of the I/Os are mapped differently, then the layout is invalidated and the unassigned pins identified.

Importing Files

Importing source files

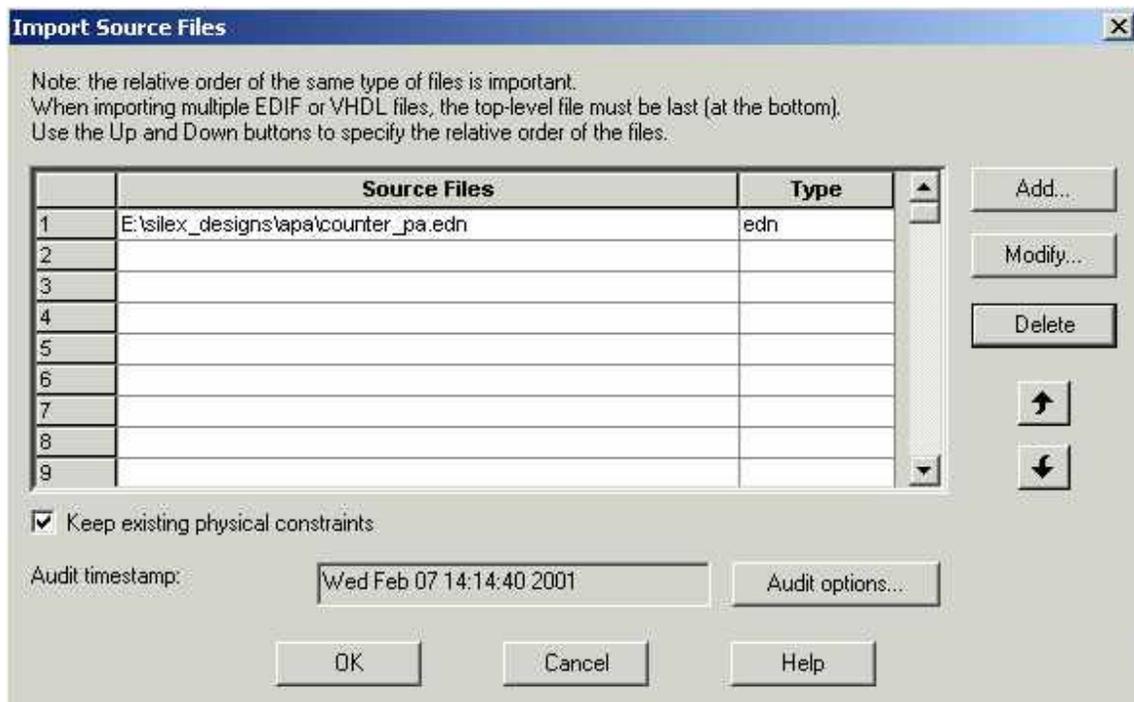
Source files include your netlist and constraint files.

Source Files	File Type Extension
EDIF	*.ed*
Verilog	*.v
VHDL	*.vhd
Actel ADL Netlist	*.adl
Criticality	*.crt
ProASIC Constraint File	*.gcf
Physical Design Constraint File	*.pdc

The choice of source files is family dependent. Only supported source files are displayed in the Import Source dialog box. If you are working on a new design or if you have changed your netlist, then you must re-import your netlist into Designer.

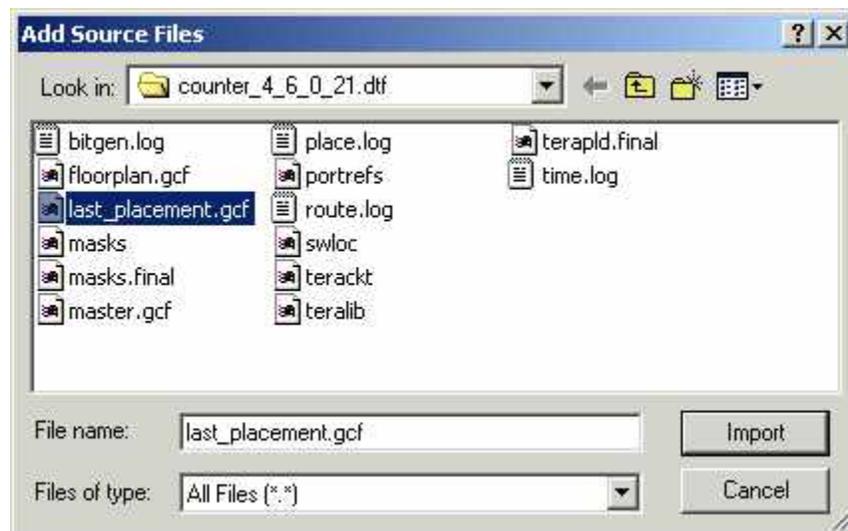
To import a source file:

1. In the **File** menu, click **Import Source Files**.



Import Source Files Dialog Box

2. Click **Add**. The Add Source Files dialog appears.



Add Source Files Dialog Box

3. Select the file you want to import and click **Import**. The File is added to the Import Source Files dialog box.
4. Add more source files to the list. All files added to the Import Source Files dialog box are imported at the same time. To modify a file, select the file and click **Modify**. To delete a file, select the file and click **Delete**.

5. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files. When importing multiple EDIF or structural HDL files, the top-level file must appear last in the list (at the bottom). Drag your files to specify the import order.
6. (ProASIC and ProASIC ^{PLUS} designs only) Select **Keep existing physical constraints** to preserve all existing physical constraints that you have made using ChipPlanner, PinEditor, or the I/O Attribute Editor. If you import a GCF file and you have this box selected, the existing physical constraints take precedents over the physical constraints in the GCF file.
7. To set the audit options for these source files, click **Audit options** and follow the directions in the Audit Options dialog box.
8. When you are done adding all your source files, click **OK**. Your source files are imported. Any errors appear in the Designer log window.

Note:

- File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.

Auditing files

Designer audits your source files to ensure that your imported source files are current. All imported source files are date and time stamped. Designer notifies you if the file is changed. When notified, select the appropriate action and click **OK**.

To change your audit settings:

1. From the **File** menu, click **Audit Settings**. The Audit Settings dialog box appears. Audit Timestamp reflects the last time and day that the import source or audit update was successfully done.

2. Select the audit check box next to the file to enable auditing.
3. Click **Change Location** to move the file to another directory.
4. Click **Reset to Current Date Time** to associate the file with the current day and time.

Importing Constraint (Auxiliary) Files

The following constraint file types can be imported into Designer:

Auxiliary Files	File Type Extension	Family
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC ^{PLUS}
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC ^{PLUS}
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

To import an auxiliary file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog appears,
2. Click the **Add** button. The Add Auxiliary Files dialog box appears.
3. Select your file and click **Import**. The file is added to the Import Auxiliary Files dialog box. Continue to add more auxiliary files to the list.
 - **Modifying:**If you need to modify a selection, select the file row and click **Modify**
 - **Deleting:**If you need to delete a file, select the file row and click **Delete**.
 - **Ordering:**Ordering your source files. Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files.
4. After you are done adding all your Auxiliary files, click **OK**. Your auxiliary files are imported. Any errors appear in Designer's Log Window.

Note:

1. .vcd and .saif are used by SmartPower for power analysis.
2. .crt for backwards compatibility with existing designs only.
3. File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.

Importing PDC files (Axcelerator family only)

The Physical Design Constraint (PDC) file can specify:

- I/O standards and features
- VCCI and VREF for all or some of the banks
- Pin assignments

- Placement locations
- Net criticality

The Axcelerator family of devices supports multiple I/O standards (with different I/O voltages) in a single die. You can use ChipEditor and PinEditor to set I/O standards and attributes, or alternatively you can export and import this information using a PDC file. PDC files are only supported for the Axcelerator family of devices.

To import a PDC file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog appears.
2. Click the **Add** button. The Add Auxiliary Files dialog box appears. Filter for your PDC file by selecting Physical Design Constraint Files (*.pdc) from the Files of Type drop-down list box.
3. Select the PDC file and click **Import**. The file is added to the Import Auxiliary Files dialog box.
4. Click **OK**. The PDC file is imported into Designer. Any errors appear in the Log Window.

Note:

- File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.
- If the PDC file has commands to combine I/O Registers with I/Os this file must be imported before compile

Importing Synopsys Design Constraint files

SDC is a Tcl-based format-constraining file. The commands of an SDC file follow the Tcl syntax rules. Designer accepts an SDC constraint file generated by a third-party tool.

To import an SDC file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog box is displayed.
2. Click **Add**. The Add Auxiliary Files dialog box appears.
3. Select your SDC file. Filter for SDC files by selecting SDC Files in the Files of Type drop-down list box.
4. Click **Import**. The SDC file is added to the Import Auxiliary Files dialog box.
5. Click **OK**. The SDC file is imported into your design. Any errors appear in the Log Window.

When Compile and Layout complete and Timer starts, the constraints from the SDC file are incorporated in the timing of the design and are reflected in Timer.

Note: File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.

Compile

Compiling your design

After you import your netlist files and select your device, you must compile your design. Compile contains a variety of functions that perform legality checking and basic netlist optimization. Compile checks for netlist errors (bad connections and fan-out problems), removes unused logic (gobbling), and combines functions to reduce logic count and improve performance. Compile also verifies that the design fits into the selected device.

There are three ways to select the compile command:

- In the Tools menu, click **Compile**.
- Click the **Compile** button in the Design Flow.

- Click the **Compile** icon in the toolbar.

If you have not already done so, Designer's Device Selection Wizard prompts you to set the device and package.

During compile, the message window in the Main window displays information about your design, including warnings and errors. Designer issues warnings when your design violates recommended Actel design rules. Actel recommends that you address all warnings, if possible, by modifying your design before you continue.

If the design fails to compile due to errors in your input files (netlist, constraints, etc.), you must modify the design to remove the errors. You must then re-import and re-compile the files.

After you compile the design, you can run Layout to place-and-route the design or use the User Tools (PinEditor, ChipEditor, ChipPlanner, Timer, SmartPower, or NetlistViewer) to perform additional optimization prior to place-and-route.

Compile Options

The compile options are specific to each family. Compile options are not available for the ProASIC and ProASIC ^{PLUS} families.

To set compile options:

1. From the **Options** menu, click **Compile**. The compile option dialog box opens.
Options available are family specific.
2. Select your options and click **OK**.

Netlist Pin Properties Overwrite Existing Properties

During the Compile process, Designer checks the netlist properties. If the netlist file specifies a pin assignment for a pin that was also assigned in PinEditor session, there is a conflict. How this conflict is resolved is determined by your selection in this box.

If this option is off, or unchecked, then Designer uses the assignment made in PinEditor and the assignment in the netlist file for the conflicting pin is ignored.

If this option is on, or checked, then Designer uses the assignment in the netlist file for that pin and the PinEditor assignment is ignored. If you edit pin assignments in PinEditor, this option is automatically set to "off."

Combine Registers into I/Os (Axcelerator Family)

The Axcelerator family includes an optional register on the input path, an optional register on the output path, and an optional register on the 3-state control pin.

Select the option Combine Registers into I/Os where possible to take advantage of these registers.

Abort on PDC Error (Axcelerator Family)

Setting Abort on PDC Error aborts the PDC import when an error is encountered. When this box is checked, the PDC file is either imported fully or the design is left untouched.

Fanout Messages (ACT1, ACT2, ACT3, DX, MX, SX, SX-A, eX)

Use the control slider in the Messages area to control the warning level. Use the control slider to specify the fanout limit that the Compile step checks against. Setting the control slider to '0' informs the system to use the system defaults. Any non-zero value replaces the system default value for the fanout limit with the user-specified value. Typically, this value range is 1 to 24.

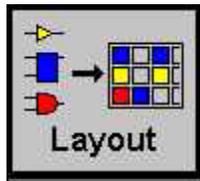
This does not adjust the fanout of the design and it has no effect on the netlist. This only adjusts the warning level, by controlling what level of fanout checking you want to be warned about during Compile. Changing this fanout limit option does not invalidate the Compile design state

Running Layout

Use Layout to place and route your design.

To run Layout:

1. Click the **Layout** button in the Design Flow Window.



2. **Layout Options.** Select your Layout options and Click **OK**. Layout options are family specific.

Layout options

Axcelerator Layout Options

When running Layout, use the Layout dialog box to set your Layout options.

Timing-driven

Select this option to run timing-driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. timing-driven Layout is more precise and typically results in higher performance.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if timing-driven Layout is required.

Run Place

Select this option to run the placer during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Place is checked by default. If your design has already been placed, this box is not checked. You can also select the following incremental placement options.

- **Incrementally:** Select to use previous placement data as the initial placement for the next placement run.
- **Lock Existing Placement (fix):** Select to use and fix previous placement data for the next incremental placement run.

Effort Level

Use the Effort Level slider to increase the effort Layout uses to place and route your design. The range is 1 to 5 with a default of 3. A higher level of effort generally improves the quality of results, but runs longer.

Run Route

Select to run the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Route is checked. Run Route is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this box is checked.

Use Multiple Passes

Select to run layout multiple times with different placement seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

Note: To run Multiple Passes, you must check both Run Place and Run Route.

Flash Layout Options

When running layout, use the Layout dialog box to set your layout options.

Timing-driven

Select this option to run timing-driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. Timing-driven Layout is more precise and typically results in higher performance.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if timing-driven Layout is required.

Run Place

Select this option to run the placer during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Place is checked by default. If your design has already been placed but not routed, this box is not checked by default. You can also select the following incremental placement options.

- **Incrementally:** Select to use previous placement data as the initial placement for the next place run.
- **Lock Existing Placement (fix):** Select to preserve previous placement data during the next incremental placement run.

Run Route

Select to run the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Route is checked. Run Route is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this box is checked.

- **Incrementally:** Select to fully route a design when some nets failed to route during a previous run. You can also use it when the incoming netlist has undergone an E.C.O. (Engineering Change Order). Incremental routing should only be used if a low number of nets fail to route (less than 50 open nets or shorted segments). A high number of failures usually indicates a less than optimal placement (if using manual placement through macros, for example) or a design that is highly connected and does not fit in the device. If a high number of nets fail, relax constraints, remove tight placement constraints, or select a bigger device and rerun routing.

Use Multiple Passes

Select to run layout multiple times with different seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

eX, SX, SX-A Layout Options

When running layout, use the Layout dialog box to set your layout options.

Timing-Driven

Select to run Timing-Driven Layout. The primary goal of Timing-Driven layout is to meet timing constraints, while still producing high performance for the rest of the design. Timing-Driven Layout is more precise and typically results in higher performance. This option is available only when timing constraints have been defined.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if Timing-Driven Layout is

required.

Place Incrementally

Select to use previous placement data as the initial placement for the next place run.

- **Lock Existing Placement:** Select to preserve previous placement data during the next incremental placement run.

Use Multiple Passes (eX and SX-A only)

Select to run layout multiple times with different seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

Advanced

Click the Advanced button to set Extended Run and Timing-Driven options.

eX, SX, and SX-A Advanced Layout Options

To set these advanced options during Layout, click the Advanced button in the Layout dialog box.

Extended Run

Select this to run a greater number of iterations during optimization within a single layout pass. An extended run layout can take up to 5 times as long as a normal layout.

Effort Level

This setting specifies the duration of the timing-driven phase of optimization during timing-driven Layout. Its value specifies the duration of this phase as a percentage of the default duration. This option is available only when timing constraints have been defined

The default value is 100 and the selectable range is within 25 - 500. Reducing the effort level also reduces the run time of timing-driven place-and-route (TDPR). With an effort

level of 25, TDPR is almost four times faster. With fewer iterations, however, performance may suffer. Routability may or may not be affected. With an effort level of 200, TDPR is almost two times slower. This variable does not have much effect on timing.

Timing Weight

Setting this option to values within a recommended range of 10-150 changes the weight of the timing objective function, thus influencing the results of timing-driven place-and-route in favor of either routability or performance. This option is available only when timing constraints have been defined

The timing weight value specifies this weight as a percentage of the default weight (i.e. a value of 100 has no effect). If you use a value less than 100, more emphasis is placed on routability and less on performance. Such a setting would be appropriate for a design that fails to route with TDPR. In case more emphasis on performance is desired, set this variable to a value higher than 100. In this case, routing failure is more likely. A very high timing value weight could also distort the optimization process and degrade performance. A value greater than 150 is not recommended.

Restore Defaults

Click **Restore Defaults** to run the factory default settings for advanced options.

ACT, MX, and DX Layout Options

Timing-driven

Select this option to run Timing-Driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. timing-driven Layout is more precise and typically results in higher performance. This option is available only when timing constraints have been defined.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if Timing-Drive Layout is required.

Place Incrementally

Select to use previous placement data as the initial placement for the next place run.

- **Lock Existing Placement:** Select to preserve previous placement data during the next incremental placement run.

Advanced

Click Advanced to set Extended Run options.

ACT, MX, and DX Advanced Layout Options

To set these advanced options during Layout, click the Advanced button in the Layout dialog box.

Extended Run

Select this to run a greater number of iterations during optimization. An extended run layout can take up to 5 times as long as a normal layout

Restore Default

Click **Restore Defaults** to run the factory default settings for advanced options.

Incremental Placement

In either standard or timing-driven mode, use incremental placement to preserve the timing of a design after a successful place and route, even if you change part of the netlist. Incremental placement has no effect the first time you run layout. During design iteration, incremental placement attempts to preserve the placement information for any unchanged macros in a modified netlist.

As a result, the timing relationships for unchanged macros approximate their initial values, decreasing the execution time to perform Layout. By forcing Designer to retain the placement information for a portion of the design, some flexibility for optimal design layout may be lost. Therefore, do not use incremental placement to place your design in pieces. You should only use it if you have successfully run Layout and you have minor changes to your design.

Incremental placement requires prior completion of place. Do not use incremental placement if the previous Layout failed to meet performance goals.

Locking Existing Placement (Fix)

When this option is selected in the Layout dialog box, all unchanged macros are treated as fixed placements during an incremental placement. This is the strongest level of control, but it may be too restrictive for the new placement to successfully complete. The default ON setting treats unchanged macro locations as placement hints, but alters their locations as needed to successfully complete placement. Refer to ChipEditor for details on fixing macros.

Flash Placement Constraint File (GCF)

For Flash designs, the Designer software always produces a placement constraints file in the design directory called: <design>.dtf/Last_placement.gcf. This file contains all the information about the latest placement. Blocks with fixed placement constraints generate fixed placement constraints, while the others generate initial placement constraints. The existing constraint files can be edited to remove any prior placement constraints. The GCF command

```
read "last_placment.gcf";
```

can be added to an existing constraint file to indicate that the latest placement is to be used as the initial placement.

Move or copy "last_placment.gcf" to use it as an input constraint file. Otherwise, it is overwritten by any subsequent placement if it is left in its original location.

Multiple Pass Layout

Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of Layout results. This is done by running individual place and route multiple times with varying placement seeds and measuring the best results with a specified criteria.

Note:

- Before running Multiple Pass Layout, you need to save your design.
- Multiple Pass Layout is supported in the following families: Axcelerator, ProASIC, ProASIC^{PLUS}, SX-A, and eX.
- Multiple Pass Layout saves your design file with the pass that has the best layout results. A corresponding timing report file for the best result, named design-name_timing.rpt is also saved to disk. If you want to preserve your existing design state, you should save your design file with a different name before proceeding. To do this, from the File menu, click Save As.

- A timing report for each pass will be written out to the working directory to assist you in later analysis. The report files will be named design-name_timing.rpt.pass-number. Look at the design-name_iteration_summary.rpt for details of the saved files.

To configure your multiple pass options:

1. When running Layout, select **Use Multiple Passes** in the Layout Options dialog box.
2. Click **Configure**. The Multi-Pass Configuration dialog box appears.
3. Set the options and click **OK**.

Maximum Number of Passes: Set the number of passes (iterations) using the slider. 3 is the minimum and 25 is the maximum. The recommended number of passes is 5.

Measurement: Select the measurement criteria you want Layout to meet. If Slowest Clock or Specific Clock is selected as your criterion, then the Layout runs all passes. If Timing Violations is selected as your criterion, Layout stops once the timing constraints are met. If the constraints are not met, then all of the Layout passes run.

Slowest Clock	Select to use the slowest clock in the design in a given pass as the performance reference for the layout pass.
Specific Clock	Select to use a specific clock as the performance reference for all Layout passes.
Timing Violations	Select to use the pass that best meets the slack or timing-violations constraints. NOTE: You must enter your own timing constraints through the Timer or SDC. The 'best' case is calculated by determining the total negative slack for all constraints.

Save Results from All Passes: Select to save the design file (.adb) for each pass. By default, only the best result is saved to your design. With this option, for every pass, the individual .adb is stored as filename_pass-number.adb in the name. The 'best' pass design will still also be written back to the original .adb filename. Saving all results does take more disk space, but allows you to later analyze the result of each pass in more detail. Look at the design-name_iteration_summary.rpt for details of the saved files.

Back-Annotation

Back-Annotation

The back-annotation functions are used to extract timing delays from your post layout data. These extracted delays are put into a file to be used by your CAE package's timing simulator. If you wish to perform pre-layout back-annotation, select Export and Timing Files from the File menu.

The Back-Annotation program creates the files necessary for back-annotation to the CAE file output type that you chose. Refer to Actel Interface Guides or the documentation included with your simulation tool for information about selecting the correct CAE output format and using the back-annotation files.

To back-annotate your design:

1. From the **Tools** menu, click **Back-Annotate**, or click the Back-Annotate button in the Design Flow window.
2. Make your selections in the Back-Annotated dialog box and click **OK**.

Extracted Files Directory: The file directory is your default working directory. If you wish to save the file elsewhere, click Browse and specify a different directory.

Extracted File Names: This name is used as the base-name of all files written out for back-annotation. Do not use directory names or file extensions in this field. The file extensions will be assigned based on your selection of which file formats to export. The default value of this field is <design>_ba.

Output Formats: Select SDF or STF (not supported for SX-A, eX, Axcelerator, ProASIC, and ProASIC ^{PLUS}).

Simulator Language: Select either Verilog or VHDL93.

Export Additional Files: Check Netlist or Pin to export these files at the same time.

Exporting files

Exporting files

Designer supports the exporting of the following file types:

Files	File Extension	Family
Actel Flattened Netlist	.afl	All
Actel Internal Netlist	.adl	All
Standard Delay Format	.sdf	All
Standard Timing File	.stf	ACT1, ACT2, ACT3, MX, XL, DX, SX
STAMP	.mod, .data	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Tcl Script File	.tcl	All
Verilog Netlist	.v	All
VHDL Netlist	.vhd	All
EDIF Netlist File	.edn	All
Log File	.log	All
STAPL	.stp	ProASIC, ProASIC ^{PLUS}
Bitstream	.bit	ProASIC, ProASIC ^{PLUS}

Data I/O Programming File (Legacy)	.dio	ACT1, ACT2, ACT3, XL, DX
Programming File (Legacy)	.fus	ACT1, ACT2, ACT3, MX, XL, DX
Actel Programming File	.afm	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Routing Segmentation File	.seg	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
Silicon Explorer Probe File	.prb	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Placement Location File	.loc	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
ProASIC Constraints File	.GCF	ProASIC, ProASIC ^{PLUS}
Combiner Info	.cob	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Boundary Scan File	.bsd	DX, 42MX, SX, SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}

Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC ^{PLUS}
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC ^{PLUS}
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Designer does not support VHDL 87 in export.

To export a file:

1. In the **File** menu, click an export option from the **Export** sub-menu. Select the type of file you wish to export.
2. Specify file name and file type and click **OK**.

Generating Reports

Available Report Types

Select from Report Type & Options on the dialog box when invoking a report.

Status Report provides information about Designer, Device Data, and variable settings for the design.

Timer Report displays summarized timing delays for paths. When Timing Report is selected and "OK" clicked, an additional dialog box prompts the user to select Timing Report Options.

Timing Violations Report summarizes timing violations.

Pin Report can be sorted by Name or Number. Select Pin Report, then click "OK" to set the List order.

FlipFlop Report can be Summary or Extended. Both reports include the Flip-Flop type, sequential (Seq) or combinatorial (CC), the Library name, and the Total number of Seq and CC Flip-Flops in the design. The Summary Report also includes the Number of instances of each unique type. The Extended Report provides the Macro name. All Reports are output to an editable window for viewing, modification, saving, and printing.

Power Reports allow you to quickly determine if any consumptions problems exist in your design.

Status Reports

The status report enables you to create a report containing device and design information, such as die, package, percentage of the logic and I/O modules used, etc.

To generate a status report:

1. In the Tools menu, click Reports.
2. Choose Status from the drop-down list in the Report Type dialog box. The status report opens in a separate window. You can save or print the report.

Timing reports

The timing report enables you to quickly determine if any timing problems exist in your design. The timing report lists the following information about your design:

- maximum delay from input I/O to output I/O
- maximum delay from input I/O to internal registers
- maximum delay from internal registers to output I/O
- maximum delays for each clock network
- maximum delays for interactions between clock networks

To generate a timing report:

1. In the **Tools** menu, click **Reports**.
2. Choose **Timing** from the Report Type drop-down list. This displays the Timing Report dialog box.
3. Specify the Slack Threshold. If you select “Slack” as the sort method, you can limit the number of delays displayed based upon a slack threshold. For example, if you only want to see delays that have a slack less than 5ns, enter 5 in the Slack Threshold box.
4. Setup-hold Timing Check. Selection of this box enables you to configure the timing report to calculate external setup and hold information for device inputs in addition to the standard information.
5. Expand Failed Paths. If a path does not meet your timing specifications, and you would like to see the incremental delay of each macro within that path, select the Expand Failed Paths box.
6. Options. Clicking Options brings up the Timing Preferences dialog box, where you can set additional display and report options.

Sort by Actual Delay

The actual delay is the path delay between two points in your design. This is the only way to sort your data if you do not have any timing constraints entered (for information on setting timing constraints, see the Timer User's Guide). If you have entered timing constraints, the actual delay report will automatically display the slack - even if you don't ask for it - but the data will always be listed from longest to shortest actual delay.

Actual delay measurements may be calculated before or after layout (that is, pre-layout or post-layout).

Sort by Slack Delay

Slack delay is the delay difference between a timing constraint entered in Timer and the actual delay of a path. For example, if a signal takes 20 ns to get from point

A to point B, and you entered a timing constraint of 15 ns, the Timing Report would list -5 ns slack for that path. Thus, if the slack negative, then the actual delay did not meet the desired timing by the absolute value of the slack (in ns). Conversely, if the slack value is positive, then the timing constraint was met, with the slack value (in ns) to spare. In a slack report, the data will be sorted (by default) from longest to shortest slack.

When displaying slack, all the paths without timing constraints are filtered from the reported data. This allows you to quickly determine how well your design meets your timing requirements. This is especially useful for viewing critical delays like register-to-register, clock-to-out, and input-to-register.

Path Selection

Normally, only the longest path between any of the starting points (terminals) and each ending terminal is displayed. If you would like to see the timing of all paths between any of the starting terminals and any of the ending terminals, select Paths Between Any Pair in the Path Selection box.

Break Path at Register

The default timing paths break at all clock, gate, clear, and preset pins. If you would like to generate a timing report that passes through these pins, unselect the appropriate pins in the Break Path at Register options.

7. Click **OK**. This displays a timing report based upon your timing and display preferences. The format and content of the report is determined by the family

Pin reports

The pin report allows you to create a text list of the I/O signal locations on a device. You can generate a pin report sorted by I/O signal names or by package number.

To generate a pin report:

1. In the **Tools** menu, click **Reports**.

2. Choose **Pin** from the drop-down list in the Report Type dialog box. This displays the Pin Report dialog box.
3. Specify the type of report to generate. Select Number or Name from the List By pull-down menu, then click **OK**. This displays a pin report.

Flip-flop reports

The flip-flop report enables you to create a report that lists the number and type of flip-flops (sequential or CC, which are flip-flops made of 2 combinatorial macros) used in a design.

There are two types of reports you can generate, Summary or Extended:

A Summary report displays whether the flip-flop is a sequential, I/O sequential, or CC flip-flop, the macro implementation of the flip-flop, and the number of times the implementation of the flip-flop is used in the design.

An Extended report individually lists the names of the macros in the design.

To generate a flip-flop report:

1. In the **Tools** menu, click **Reports**. This displays the Reports dialog box.
2. Select Flip-Flop from the drop-down menu. The Flip-Flop Report dialog box appears.
3. Specify the type of report to generate. Select Summary or Extended from the Type pull-down menu, then click **OK**. This displays the report in a separate window.

Power reports

The power report enables you to quickly determine if any power consumption problems exist in your design. The power report lists the following information:

- Global device information and SmartPower Preferences selection information
- Design level static power summary
- Dynamic power summary
- Hierarchical detailed power report (including gates, blocks, and nets), with a block by block, gate by gate, and net by net power summary SmartPower results

To create a power report:

1. In the **Tools** menu, click **Reports**. This displays the Reports dialog box.
2. Choose **Power** in the Report list and click **OK**. The Power Report dialog appears.
3. Choose from the following options:
 - **Static Power**: Returns static power information
 - **Dynamic Power**: Returns dynamic power information
 - **Report Style**: Specifies report style

For additional Power Report Options, click the Options to open the Power Preferences dialog box, as shown in Figure 2-45.

Power Preferences Dialog Box

Select analysis preferences:

- **Units**: Sets units preferences for power and frequency
- **Operating Conditions**: Sets preferences for operating conditions
- **Block Expansion Control**: Filters reported power values returned in the report.

This box does not control which values are included, rather it specifies which blocks are detailed/expanded. You may specify which blocks are expanded using a minimum power value, a minimum power ratio (with regards to the total power of the design) and a maximum hierarchical depth; a filtered value is not include in displayed lists, but still counted for upper hierarchical levels.

4. Once you are satisfied with your selections, click **OK** in the Preferences dialog box and then click **OK** in the Power Report dialog box. SmartPower displays the report in a separate window.

Timing Violations Reports

For families that use the pin-to-pin timing model, the Violations report enables you to

obtain constraint results sorted by slack. You can now view Max Delay violations as well as Min Delay violations in the report.

To generate a timing violations report:

1. From **Tools** menu, click **Reports**.
2. In the Report Types dialog box, select **Timing Violations**.
3. Click **OK**.

Saving and Exiting

Saving your design

Once you have imported a netlist and compiled a design, you can save the design as an ADB file.

To save your design as an ADB file:

1. In the **File** menu, click **Save** or click the save icon in the toolbar.
2. Enter the File name and click **Save**. The default file name is the name you previously entered in the setup dialog box. The default format is adb. Make sure your save in the “.adb” format.

Once you have saved your compiled design as an ADB file, during any future Designer sessions, you can open the ADB file, skipping the compile step, and perform optimization on the design, including updating netlist and auxiliary file information.

Exiting Designer

To end a Designer session, from the **File** menu, click **Exit**.

If the information has not been saved to disk, you are asked if you want to save the design before exiting. If you choose YES, the "<design_name>.adb" file is updated with information entered the current session. If you choose NO, the information is not saved and the "<design_name>.adb" file remains unchanged.

Tcl Scripting

Tcl Documentation Conventions

The Actel command syntax conventions are as follows.

Syntax Notation	Description
command	Commands or keywords are shown in <code>courier</code> typeface.
<i>Variable</i>	Variables appear in italic. You must substitute an appropriate value for the variable.
?argument?	Optional argument. Do not use the question marks when entering the argument.
arg1 arg2 ... argN	Alternative arguments. You can use exactly one of these arguments.
...	The ellipsis indicate items that precede the ellipsis may be repeated. The ellipsis should not be entered.

Tcl Command Reference

Designer supports the following Tcl scripting commands.

- backannotate
- close_design
- compile

- export
- get_defvar
- get_design_filename
- get_design_info
- is_design_loaded
- is_design_modified
- is_design_state_complete
- layout (advanced options for the SX family)
- new_design
- open_design
- pin_assign
- pin_commit
- pin_fix
- pin_fix_all
- pin_unassign
- pin_unassign_all
- pin_unfix
- save_design
- set_defvar
- set_design
- set_device
- smartpower_add_pin_in_domain

- smartpower_commit
- smartpower_create_domain
- smartpower_remove_domain
- smartpower_remove_pin_frequency
- smartpower_remove_pin_of_domain
- smartpower_restore
- smartpower_set_domain_frequency
- smartpower_set_pin_frequency
- timer_add_clock_exception
- timer_add_pass
- timer_add_stop
- timer_commit
- timer_get_path
- timer_get_clock_actuals
- timer_get_clock_constraints
- timer_get_maxdelay
- timer_get_path_constraints
- timer_remove_clock_exception
- timer_remove_pass
- timer_remove_stop
- timer_restore
- timer_setenv_clock_freq

- timer_setenv_clock_period
- timer_set_maxdelay
- timer_remove_all_constraints

backannotate

The `backannotate` command is equivalent to executing the Back-Annotate command within the Tools menu. You can export an SDF file, after layout, along with the corresponding netlist in the VHDL or Verilog format. These files are useful in backannotated timing simulation.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
Backannotate ?Option Option ...?
```

```
-name file_name  
-format format_type  
-language  
-simlang  
-dir dir  
-netlist  
-pin
```

Arguments

```
?-name file_name?
```

Use a valid file name with this option. You can attach the file extension `.sdf` to the `File_Name`, otherwise the tool will append `.sdf` for you.

```
?-format format_type?
```

Only SDF format is available for back annotation

```
?-language language?
```

The supported Languages are options are

VHDL93 – For VHDL-93 style naming in SDF

VERILOG – For Verilog style naming in SDF

```
?-dir directory_name?
```

Specify the directory in which all the files will be extracted.

```
?-netlist?
```

Forces a netlist to be written. The netlist will be either in Verilog or VHDL depending on the

```
-language option.
```

The netlist file name will have the appropriate extension `.v` or `.vhd` appended to reflect the netlist format.

?-pin?

Designer exports the pin file with this option. The `.pin` file extension is appended to the design name to create the pin file.

Notes

We advise you to export both SDF and the corresponding VHDL/Verilog files. This will avoid name conflicts in the simulation tool.

Designer must have completed layout before this command can be invoked, otherwise the command will fail.

Exceptions

-pin is not supported for ProASIC and ProASICPLUS families.

Example

Example 1:

```
backannotate
Uses default arguments and exports SDF file for back annotation
```

Example 2:

```
backannotate -dir \
{..\my_design_dir} -name "fanouttest_ba.sdf" -format "SDF" -
language \ "VHDL93" -netlist
This example uses some of the options for VHDL
```

Example 3:

```
backannotate -dir \
{..\design} -name "fanouttest_ba.sdf" -format "SDF" -language
"VERILOG" \
-netlist
This example uses some of the options for Verilog
```

Example 4:

```
If { [catch { backannotate -name "fanouttest_ba" -format "SDF"
} ]} {
    Puts "Back annotation failed"
    # Handle Failure
} else {
    Puts "Back annotation successful"
    # Proceed with other operations
}
```

You can catch exceptions and respond based on the success of backannotate operation

close_design

The `close_design` command closes the current design and brings Designer to a fresh state to work on a new design.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
close_design
```

Arguments

None.

Notes

This is equivalent to selecting the Close command in the File menu.

Exceptions

None.

Example

```
if { [catch { close_design }] } {  
    Puts "Failed to close design"  
    # Handle Failure  
}  
else {  
    puts "Design closed successfully"  
    # Proceed with processing a new design  
}
```

See Also

`open_design`, `close_design`, `new_design`

compile

The compile command performs design rule check on the input netlist. If the compile is successful, Designer reaches the compiled state. Compile also performs some optimizations on the design through logic combining and buffer tree modifications.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
compile ?argument argument ...?
```

Arguments

?-combine_register value?.

Combines registers at the IO into IO-Registers. The value should be 1 for this optimization to take effect.

?-nl_pins_overwrite?

This option is used to overwrite the imported netlist with the changes made in PinEditor.

Notes

-combine_register option is available only for Axcelerator family.
-nl_pins_overwrite option is not available for Axcelerator, ProASIC and ProASIC^{PLUS}.

Exceptions

There are no compile options available for ProASIC and ProASIC^{PLUS}.

Example

Example 1:

```
compile -combine_register
```

Example 2:

```
if { [catch { compile -nl_pins_overwrite }] } {  
    Puts "Failed compile"  
    # Handle Failure  
} else {  
    puts "Compile successful"  
    # Proceed to Layout  
}
```

export

The export command can be used to create a variety of files from Designer. The user through appropriate format and options can select these files for export. The basic types of files supported are listed in the following table.

Files	File Type Extension	Family
Actel Flattened Netlist	.afl	All
Actel Internal Netlist	.adl	All
Standard Delay Format	.sdf	All
Standard Timing File	.stf	ACT1, ACT2, ACT3, MX, XL, DX, SX
STAMP	.mod, .data	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Tcl Script File	.tcl	All
Verilog Netlist	.v	All
VHDL Netlist	.vhd	All
EDIF Netlist File	.edn	All
Log File	.log	All
STAPL	.stp	ProASIC, ProASIC ^{PLUS}
Bitstream	.bit	ProASIC, ProASIC ^{PLUS}
Data I/O Programming File (Legacy)	.dio	ACT1, ACT2, ACT3, XL, DX
Programming File (Legacy)	.fus	ACT1, ACT2, ACT3, MX, XL, DX
Actel Programming File	.afm	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Routing Segmentation File	.seg	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Silicon Explorer Probe File	.prb	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Placement Location File	.loc	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
ProASIC Constraints File	.GCF	ProASIC, ProASIC ^{PLUS}
Combiner Info	.cob	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Boundary Scan File	.bsd	DX, 42MX, SX, SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC ^{PLUS}
Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC ^{PLUS}
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC ^{PLUS}
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Supported Family and Format

Family: All

Format: TCL

Syntax

```
export -format edif \  
    -edif_flavor ( generic | viewlogic | mgc | orcad | workview ) \  
                {filename}  
export -format ( afm | dio | fus ) [-signature value] {filename}  
export -format log -diagnostic {filename}  
export -format sdf [-prelayout] {filename}  
export -format ( adl | afl | cob | crt | dcf | design_script | loc |  
pin | session_script | stf | tcl | verilog | vhd1 | crt | dcf )  
                {filename}
```

get_defvar

The `get_defvar` command provides access to the internal variables within Designer and returns its value.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

`get_defvar variable`

Arguments

The *variable* is the Designer internal variable.

Notes

This command also prints the value of the Designer variable on the log window.

Exceptions

None.

Example

Example 1: Prints the design name on the log window.

```
get_defvar "DESIGN"  
set variableToGet "DESIGN"  
set valueOfVariable [get_defvar $variableToGet]  
puts "The value is $valueOfVariable"
```

See Also

`set_defvar`

get_design_filename

The `get_design_filename` command can be used to retrieve the full qualified path of the design file.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
get_design_filename
```

Arguments

None.

Notes

- The result will be an empty string if the design has not been saved to disk.
- This command is equivalent to the command “`get_design_info` DESIGN_PATH.” This command predates `get_design_info` and is supported for backward-compatibility.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if arguments are passed.

Example

```
if { [ is_design_loaded ] } {  
    set design_location [ get_design_filename ]  
    if { $design_location != "" } {  
        puts "Design is at $design_location."  
    } else {  
        puts "Design has not been saved to a file on disk."  
    }  
} else {  
    puts "No design is loaded."  
}
```

See Also

`get_design_info`
`is_design_loaded`
`is_design_modified`

is_design_state_complete

get_design_info

The `get_design_info` command can be used to retrieve some basic details of your design.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
get_design_info name | family | design_path | cwdir | die | package  
| speed | design_state
```

Arguments

The single argument must be one of the valid string values.

`name`

Design name. The result is set to the design name string.

`family`

Silicon family. The result is set to the family name.

`design_path`

Full qualified path of the design file. The result is set to the location of the `.adb` file. If a design has not been saved to disk, the result will be an empty string. This command replaces the command `get_design_filename`.

`design_folder`

Directory (folder) portion of the `design_path`.

`design_file`

Filename portion of the `design_path`.

`cwdir`

Current working directory. The result is set to the location of the current working directory

`die`

Die name. The result is set to the name of the selected die for the design. If no die is selected, this is an empty string.

`Package`

Package name. The result is set to the name of the selected package for the design. If no package is selected, this is an empty string.

`Speed`

Speed grade. The result is set to the speed grade for the design. If no speed grade is selected, this is an empty string.

Notes

The result value of the command will be a string value.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if more than one argument is passed.
- The command will return an error if the argument is not one of the valid values.

Example

The following example uses `get_design_info` to display the various values to the screen.

```
if { [ is_design_loaded ] } {
    puts "Design is loaded."
    set bDesignLoaded 1
} else {
    puts "No design is loaded."
    set bDesignLoaded 0
}
if { $bDesignLoaded != 0 } {
    set var [ get_design_info NAME ]
    puts "  DESIGN NAME:\t$t$var"
    set var [ get_design_info FAMILY ]
    puts "  FAMILY:\t$t$var"
    set var [ get_design_info DESIGN_PATH ]
    puts "  DESIGN PATH:\t$t$var"
    set var [ get_design_info DESIGN_FILE ]
    puts "  DESIGN FILE:\t$t$var"
    set var [ get_design_info DESIGN_FOLDER ]
    puts "  DESIGN FOLDER:\t$t$var"
    set var [ get_design_info CWDIR ]
    puts "  WORKING DIRECTORY: $var"
    set var [ get_design_info DIE ]
    puts "  DIE:\t$t$var"
    set var [ get_design_info PACKAGE ]
    puts "  PACKAGE:\t'$var'"
    set var [ get_design_info SPEED ]
    puts "  SPEED GRADE:\t$t$var"
    if { [ is_design_modified ] } {
        puts "The design is modified."
    } else {
        puts "The design is unchanged"
    }
}
puts "get_design.tcl done"
```

See Also

`get_design_filename`

`is_design_loaded`

is_design_modified

is_design_state_complete

is_design_loaded

The `is_design_loaded` returns a Boolean value (0 for false, 1 for true) indicating if a design is loaded in the Designer software. True is returned if a design is currently loaded.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
is_design_loaded
```

Arguments

None

Notes

Some Tcl commands are valid only if a design is currently loaded in Designer. Use the `'is_design_loaded'` command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if a design has been loaded.

```
set bDesignLoaded [ is_design_loaded ]  
if { $bDesignLoaded == 0 } {  
    puts "No design is loaded."  
}
```

See Also

`get_design_filename`
`get_design_info`
`is_design_modified`
`is_design_state_complete`

is_design_modified

The `is_design_loaded` returns a Boolean value (0 for false, 1 for true) indicating if a design is loaded in the Designer software. True is returned if a design is currently loaded.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
is_design_loaded
```

Arguments

none

Notes

Some Tcl commands are valid only if a design is currently loaded in Designer. Use the `is_design_loaded` command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if a design has been loaded.

```
set bDesignLoaded [ is_design_loaded ]  
if { $bDesignLoaded == 0 } {  
    puts "No design is loaded."  
}
```

See Also

`get_design_filename`
`get_design_info`
`is_design_modified`
`is_design_state_complete`

is_design_state_complete

The `is_design_state_complete` command returns a Boolean value (0 for false, 1 for true) indicating if a specific design state is valid. True is returned if the specified design state is valid.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
is_design_state_complete SETUP_DESIGN | DEVICE_SELECTION |  
NETLIST_IMPORT | COMPILE | LAYOUT | BACKANNOTATE | PROGRAMMING_FILE
```

Arguments

The single argument must be one of the valid string values.

SETUP_DESIGN

The design is loaded and the family has been specified for the design.

DEVICE_SELECTION

The design has completed device selection (die and package). This corresponds to having successfully called the `set_device` command to set the die and package.

NETLIST_IMPORT

The design has imported a netlist.

COMPILE

The design has completed the compile command.

LAYOUT

The design has completed the layout command.

BACKANNOTATE

The design has exported a post-layout timing file (e.g. SDF).

PROGRAMMING_FILE

The design has exported a programming file (e.g. AFM).

Notes

1. Certain commands can only be used after Compile or Layout has been completed.
2. The `is_design_state_complete` command allows a script to check the design state before calling one of these state-limited commands.

Exceptions

1. The command will return an error if a design is not loaded.

2. The command will return an error if more than one argument is passed.
3. The command will return an error if the argument is not one of the valid values.

Example

The following code runs layout, but checks that the design state for layout is complete before calling backannotate.

```
layout -timing_driven
set bLayoutDone [ is_design_state_complete LAYOUT ]
if { $bLayoutDone != 0 } {
    backannotate -name {mydesign_ba} -format "SDF" -language "verilog"
}
}
```

See Also

- compile
- get_design_filename
- get_design_info
- is_design_loaded
- is_design_modified
- layout
- set_design
- set_device

layout (advanced options for the SX family)

This is equivalent to executing commands within the Advanced Layout Options dialog box.

Supported Family and Format

Family: SX

Format: Tcl

Syntax

```
layout [-timing_driven] [-incremental inc_mode] [-extended_run  
ext_mode] [-effort_level enumber] [-timing_weight tnumber]  
    where inc_mode = "on" | "off" | "fix" , ext_mode = "on" | "off" ,  
    enumber is 25 to 500, tnumber is 10-150
```

new_design

The `new_design` command creates a new design.

Supported Family and Format

Family: All

Format: TCL

Syntax

```
new_design -name design_name -family family_name -path pathname
```

Arguments

`-name design_name` .

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name` .

The Actel device family for which the design is being targeted.

`-path path_name` .

The physical path of the directory in which the design files will be created.

Notes

You need all the 3 arguments for this command. This command will setup the Designer software for importing design source files.

Exceptions

None.

Example

Example 1: Creates a new ACT3 design with the name "test" in the current folder.

```
new_design -name "test" -family "ACT3" -path {.}
```

Example 2: These set of commands create a new design through variable substitution.

```
set desName "test"
set famName "ACT3"
set path {d:/examples/test}
new_design -name $desName -family $famName -path $path
```

Example 3: Design creation and catch failures

```
if { [catch { new_design -name $desName -family $famName -path
$path }] } {
    puts "Failed to create a new design"
```

```
        # Handle Failure
    } else {
        puts "New design creation successful"
        # Proceed to Import source files
    }
```

See Also

`open_design`, `save_design`, `close_design`, `set_design`

open_design

The `open_design` command opens an existing design into the Designer software.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
open_design file_name
```

Arguments

file_name is the complete adb file path. The complete path is not provided then the directory is assumed to be the current working directory.

Notes

All previously open designs must be closed before opening a new design.

Exceptions

None.

Example

Example 1: Opens an existing design from the file "test.adb" in the current folder.

```
open_design {test.adb}
```

Example 2: Design creation and catch failures.

```
set designFile {d:/test/my_design.adb}
if { [catch { open_design $designFile } ] } {
    puts "Failed to open design"
    # Handle Failure
} else {
    puts "Design opened successfully"
    # Proceed to further processing
}
```

See Also

`new_design`, `save_design`, `close_design`

pin_assign

The `pin_assign` command assigns the pin, but does not fix its assignment.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

```
pin_assign [-nofix] -port <portname> -pin <pin number>  
pin_assign -port <port name> [-iostd <i/o standard>]  
[-iothresh <i/othreshold>][-outload <output load>]  
[-slew <High | Low>][-res_pull <None | High | Low>]
```

Arguments

-iostd
Allows to set the I/O Standard

-iothresh
Allows to set the I/O Threshold

-outload
Allows to set the Output Load, also called Loading for some families

-slew
Allows to set the Slew

-res_pull
Allows to set the Resistor Pull, also called Power Up State for some families.

Notes

Must use `pin_commit` after this command.

Exceptions

None.

pin_commit

The `pin_commit` command saves the pin assignments to the `.adb` file.

Supported Family and Format

Family: ALL

Format: Tcl

Syntax:

```
pin_commit
```

Arguments

None.

Notes

This is needed after all pin commands to save changes.

Exceptions

None.

Example

Example 1:

```
pin_commit
```

See Also

`pin_fix`, `pin_unfix`, `pin_assign`, `pin_unassign`

pin_fix

This is equivalent to fixing a pin assignment.

Supported Family and Format

Family: All
Format:Tcl

Syntax:

```
pin_fix -port port_name
```

Arguments

-port *port_name*
specifies the port name for which the pin needs to be fixed at its placed location.

Notes

Fixed pins cannot be moved during place-and-route. Must use pin_commit after this command.

Exceptions

None.

Example

Example 1:

```
Pin_fix -port {clk}  
Pin_commit
```

See Also

pin_commit, pin_unfix, pin_assign, pin_unassign

pin_fix_all

The pin_fix_all commands fixes all the placed pins on the device.

Supported Family and Format

Family: All
Format: TCL

Syntax:

```
pin_fix_all
```

Arguments

None.

Notes

Must use pin_commit after this command exceptions

Example

Example 1:

Pin_fix_all

Pin_commit

See Also

[pin_fix](#), [pin_unfix](#), [pin_commit](#), [pin_unassign](#)

pin_unassign

The `pin_unassign` command unassigns a specific pin.

Supported Family and Format

Family: All
Format: Tcl

Syntax:

```
pin_unassign -port port_name
```

Arguments

`-port port_name`
specifies the port for which the pin must be unassigned.

Notes

The unassigned pin location is now available for other ports. Must use `pin_commit` after this command.

Exceptions

Example

Example 1:

```
Pin_unassign -port "clk"  
Pin_commit
```

See Also

`pin_fix`, `pin_unfix`, `pin_commit`, `pin_unassign`

pin_unassign_all

The `pin_unassign_all` command unassigns all the pins.

Supported Family and Format

Family: All
Format: Tcl

Syntax:

```
pin_unassign_all
```

Arguments

None.

Notes

Now all the pin locations are available for assignment. Must use `pin_commit` after this command.

Exceptions

None.

Example

Example 1:

```
pin_unassign_all  
  
pin_commit
```

See Also

`pin_fix`, `pin_unfix`, `pin_commit`, `pin_unassign`

pin_unfix

The `pin_unfix` command unfixing a pin assignment, allowing it to be moved during place-and-route.

Supported Family and Format

Family: All
Format: Tcl

Syntax:

```
pin_unfix -port port_name
```

Arguments

```
-port port_name
```

specifies the port name that must be unfix.

Notes

`Pin_commit` command must be used for this command to take effect.

Exceptions

None.

Example

Example 1:

```
Pin_unfix -port "rst"  
Pin_commit
```

See Also

`pin_fix`, `pin_commit`, `pin_assign`, `pin_unassign`

report

The report command gives the user the ability to generate the Power report using TCL

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}
Format: Tcl

Syntax:

```
report -type "power" -sortby "Power Values" / "Alphabetical" -  
sortorder "Descending" / "Ascending" -style "Hierarchical" -opcond  
"Typical" -stat_pow "TRUE" / "FALSE" -dyn_pow "TRUE" / "FALSE" -domains  
"TRUE" / "FALSE" -annotated_pins "TRUE" / "FALSE" -min_ratio "number" -  
max_depth "number" -min_power "number mW" {.\report_name.rpt}
```

Arguments

- type "power"
Specifies that the type for the report to be generated is a power report
- ?-sortby "Power Values" / "Alphabetical"?
Specifies the method of sorting the values in the report
- ?-sortorder "Descending" / "Ascending"?
Specifies the sort order of the values in the report
- ?-style "Hierarchical" ?
Specifies the style of displaying the results in the report
- ?-opcond "Typical" ?
Specifies what operating conditions to be used
- ?-stat_pow "TRUE" / "FALSE"?
Specifies whether to include the Static Power value in the report.
- ?-dyn_pow "TRUE" / "FALSE"?
Specifies whether to include the Dynamic Power in the report.
- ?-domains "TRUE" / "FALSE"?
Specifies whether to include the modifies domains into the power report
- ?-annotated_pins "TRUE" / "FALSE"?
Specifies whether to include the annotated pins into the report or not
- ?-min_ratio "number" ?
Specifies which block to be expanded based on the minimum power ratio of a block with respect to the overall power value.
- ?-max_depth "number" ?
Specifies the maximum hierarchy depth to be included in the report.
- ?-min_power "number mW" ?
Specifies which block to be expanded based on the minimum power value of a block.

```
{.\report_name.rpt}
```

Specifies the name and destination of the report

Notes

None

Exceptions

None

Example

```
report -type "power" -sortBy "Power Values" -sortorder "Descending" -  
style "Hierarchical" -opcond "Typical" -stat_pow "TRUE" -dyn_pow  
"TRUE"  
-domains "TRUE" -annotated_pins "TRUE" -min_ratio "10" -max_depth "2"  
-min_power "2 mW" {e:\SmartPower\report.rpt}
```

save_design

The `save_design` command saves the current design in Designer to a file.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
save_design filename
```

Arguments

The design is written to a file denoted by the variable *filename* as an ADB file.

Notes

If *filename* is not a complete path name, the ADB file is written into the current working directory.

Exceptions

None.

Example

Example 1: Saves the design to a file "test.adb" in the current folder.

```
save_design {test.adb}
```

Example 2: Save design and check if it saved successfully.

```
set designFile {d:/test/my_design.adb}
if { [catch { save_design $designFile }] } {
    Puts "Failed to save design"
    # Handle Failure
} else {
    puts "Design saved successfully"
    # Proceed to make further changes
}
```

See Also

`open_design`, `close_design`, `new_design`

set_design

This `set_design` command specifies the design name, family and path in which Designer will process the design. This step is absolutely required before importing the source files.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_design -name design_name -family family_name -path path_name
```

Arguments

`-name design_name` .

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name` .

The Actel device family for which the design is being targeted.

`-path path_name` .

The physical path of the directory in which the design files will be created.

Notes

You need all 3 arguments for this command to setup your design.

Example

Example 1: Sets up the design and checks if there are any errors

```
set_design -name "test" -family "Axcelerator" -path {..}
set desName "test"
set famName "ACT3"
set path {d:/examples/test}

if { [catch { set_design -name $desName -family $famName -path $path } ] } {
    puts "Failed setup design"
    # Handle Failure
} else {
    puts "Design setup successful"
    # Proceed to Import source files
}
```

See Also

`new_design`, `set_device`

set_device

The `set_device` command specifies the type of device and its parameters.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_device Option1 ?Option2 Option3 ...?
```

Options:

```
?-family family_name?
?-die die_name?
?-package package_name?
?-speed speed_grade?
?-voltage voltage?
?-voltrange volt_range?
?-temprange temp_range?
?-pci yes|no?
?-jtag yes|no?
?-probe yes|no?
?-itol 3.3|5.0?
?-io_trip pci|ttl?
?-trst yes|no?
?-scope "session" |...?
?-iostd "PCIX" |...?
```

Arguments

- family *family_name*
Specifies the name of the FPGA device family.
- die *die_name*
Specifies the part name.
- package *package_name*
Specifies the selected package for the device.
- speed *speed_grade*
Specifies the speed grade of the part.
- voltage *voltage*
Specifies the core voltage of the device.
- voltrange *volt_range*
Specifies the voltage range to be applied for the device. It is generally MIL, COM and IND denoting Military, Commercial and Industrial respectively.
- temprange *temp_range*

Specifies the voltage range to be applied for the device. It is generally MIL, COM and IND denoting Military, Commercial and Industrial respectively.

`-pci yes|no`

Specified if the device needs to configure the IO for PCI specification.

`-jtag yes|no`

Specifies if pins need to be reserved for JTAG.

`-probe yes|no`

Specifies if the pins need to be preserved for probing.

`-trst yes|no`

Specifies if the pins need to be reserved for JTAG test reset.

Notes

At least one option must be specified for this command. Some of the options may not apply for certain families that do not support the features.

Example

Example 1: Setting up a PA design.

```
set_device -die "APA075" -package "208 PQFP" -speed "STD" -voltage  
"2.5" \  
-jtag "yes" -trst "yes" -temprange "COM" -voltrange "COM"
```

See Also

`new_design`, `set_design`

set_defvar

The `set_defvar` command sets an internal variable in the Designer system.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_defvar variable value
```

Arguments

Variable must be a valid Designer internal variable and could be accompanied by an optional value. If the *value* is provided, the *variable* is set the *value*. If the *value* is null the *variable* is reset.

Notes

Must have at least one argument.

Exceptions

None.

Example

Example 1:

```
set_defvar "FOREMAT" "VHDL"  
Sets the FOREMAT internal variable to VHDL.
```

Example 2:

```
set variableToSet "DESIGN"  
set valueOfVariable "VHDL"  
set_defvar $variableToSet $valueOfVariable
```

These commands set the FORMAT variable to VHDL, shows the use of variables for this command.

See Also

`get_defvar`

smartpower_add_pin_in_domain

`smartpower_add_pin_in_domain` adds a pin into a Clock or Set domain.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type {clock} /  
{set} - domain_name {domain_name} -domain_type {clock} | {set}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to be added to the domain

`-pin_type {clock} | {data}`

Specifies the type of the pin to be added. The pin added will be either a clock pin or a data pin.

`-domain_name {domain_name}`

Specifies the name of the domain to be added

`-domain_type {clock} | {set}`

Specifies the type of the domain to be added.

Notes

- The *domain_name* must be a name of an existing domain.
- The *pin_name* must be a name of a pin that exists in the design.

Exceptions

None

Example

Example 1: To a pin to an existing Clock domain

```
smartpower_add_pin_in_domain -pin_name { XCMP3/U0/U1:Y } -  
pin_type {clock} -domain_name {clk1} -domain_type {clock}
```

Example 2: To add a data pin to an existing Set domain

```
smartpower_add_pin_in_domain -pin_name {XCMP3/U0/U1:Y} -pin_type  
{data} -domain_name {myset} -domain_type {set}
```

See Also

`smartpower_remove_pin_of_domain`

smartpower_commit

The smartpower_commit command saves the changes made to the Designer database.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}
Format: Tcl

Syntax:

```
Smart_power_commit
```

Arguments

None

Notes

None

Exceptions

None

Example

```
Smart_power_commit
```

See Also

smartpower_restore

smartpower_create_domain

The `smartpower_create_domain` creates a new clock or set domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_create_domain -domain_type {clock}| {set} -domain_name  
{domain_name}
```

Arguments

`-domain_type {clock}| {set}`

Specifies that the domain that is being created is either a clock domain or set domain

`-domain_name {domain_name}`

Specifies the domain name to be created.

Notes

The `-domain_type` must be either "clock" or "set"

The domain name must not be a name of an existing domain.

Exceptions

None

Example

```
smartpower_create_domain -domain_type {clock} -domain_name {clk2}  
smartpower_create_domain -domain_type {set} -domain_name {myset}
```

See Also

`smartpower_remove_domain`

smartpower_remove_domain

The `smartpower_remove_domain` removes and existing domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_remove_domain -domain_type {clock}/ {set} -domain_name  
{domain_name}
```

Arguments

`-domain_type {clock}/ {set}`

Specifies that the domain that is being removed is either a clock domain or a set domain

`-domain_name {domain_name}`

Specifies the domain name to be removed.

Notes

- The `-domain_type` must be either “clock” or “set”
- The domain name must be a name of an existing domain.

Exceptions

None

Example

```
smartpower_remove_domain -domain_type {clock} -domain_name {clk2}  
smartpower_remove_domain -domain_type {set} -domain_name {myset}
```

See Also

`smartpower_create_domain`

smartpower_remove_pin_frequency

`smartpower_remove_pin_frequency` command removes the frequency of a pin

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{plus}

Format: Tcl

Syntax:

```
smartpower_remove_pin_frequency -pin_name {pin_name}
```

Arguments

```
-pin_name {pin_name}
```

Specifies the name of the pin for which the frequency will be removed

Notes

- The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.

Exceptions

None

Example

```
smartpower_remove_pin_frequency -pin_name {count8_clock}
```

See Also

```
smartpower_set_pin_frequency
```

smartpower_remove_pin_of_domain

`smartpower_remove_pin_of_domain` removes a clock pin or a data pin from a Clock or Set domain respectively.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_remove_pin_of_domain -pin_name {pin_name} -pin_type {data} |  
{clock} -domain_name {domain_name} -domain_type {set} | {clock}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to be removed

`-pin_type {data} | {clock}`

Specifies the type of the pin to be removed. The pin could be either a data pin or a clock pin.

`-domain_name {domain_name}`

Specifies the domain name from which to delete the pin

`-domain_type {set} | {clock}`

Specifies the type of the domain from which a pin is being removed.

Notes

- The *pin_name* must be a name of a pin that already exists in the design
- The *domain_name* must be a name of an existing domain

Exceptions

None

Example

Example 1: Removes a pin from a Clock domain

```
smartpower_remove_pin_of_domain -pin_name {XCMP3/U0/U1:Y}  
-pin_type {clock} -domain_name {clockh} -domain_type {clock}
```

Example 2: Removes a data pin from a Set domain

```
smartpower_remove_pin_of_domain -pin_name {count2_en} -pin_type  
{data} -domain_name {InputSet} -domain_type {set}
```

See Also

`smartpower_add_pin_in_domain`

smartpower_restore

The smartpower_restore command restores previous committed constraints

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

Smart_power_restore

Arguments

None

Notes

None

Exceptions

None

Example

Smart_power_restore

See Also

smartpower_commit

smartpower_set_domain_frequency

The `smartpower_set_domain_frequency` sets the frequency of a domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_set_domain_frequency -domain_type {clock}| {set} -  
domain_name {domain_name} -clock_freq {clock_frequency} -data_freq  
{data_frequenc}
```

Arguments

`-domain_type {clock}| {set}`

Specifies that the domain to set the frequency for is either a Clock domain or a Set domain

`-domain_name {domain_name}`

Specifies the domain name

`-clock_freq {clock_frequency}`

Specifies the frequency values in positive decimal number. Units in MHz.

`-data_freq {data_frequenc}`

Specifies that data frequency of the domian

Notes

- The `-domain_type` must be either “clock” or “set”
- The domain name must be a name of an existing domain.
- The clock frequency must be a positive decimal number. Specifying the unit as part of the frequency value is optional. There must be a space between the frequency value and the unit. The clock frequency needs to be set only for the clock domains.
- The data frequency must be a positive decimal number. Specifying the unit as part of the data frequency value is optional. There must be a space between the data frequency value and the unit.

Exceptions

None

Example

Example 1: Setting the clock frequency and the data frequency of a clock domain.

```
smartpower_set_domain_frequency -domain_type {clock} -domain_name  
{clk1} -clock_freq {32} or {30 MHZ} -data_freq {3} or {3 Mhz}
```

Example 2: Setting the data frequency of a set domain.

```
smartpower_set_domain_frequency -domain_type {set} -domain_name  
{set1} -data_freq {10}.
```

See Also

`smartpower_create_domain`

`smartpower_remove_domain`

smartpower_set_pin_frequency

The `smartpower_set_pin_frequency` command sets the frequency of a pin.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{PLUS}

Format: Tcl

Syntax:

```
smartpower_set_pin_frequency -pin_name {pin_name} -pin_freq  
{frequency_value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin for which the frequency will be set

`-pin_freq {frequency_value}`

Specifies the values of the frequency. The frequency can be any positive decimal number. Units in MHz

Notes

- The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.
- When specifying the unit, a space must be between the frequency value and the unit.

Exceptions

None

Example

```
smartpower_set_pin_frequency -pin_name {count8_clock} -pin_freq {100}
```

See Also

`smartpower_remove_pin_frequency`

timer_add_clock_exception

The `timer_add_clock_exception` adds an exception to or from a pin with respect to a specified clock.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_add_clock_exception -clock clock_name -pin pin_name -dir  
{from}|{to}
```

Arguments

`-clock clock_name`

Specifies the clock name.

`-pin pin_name`

Specifies the exception on the pin in a synchronous network that should be excluded from the specified clock period.

`-dir {from}|{to}`

Specifies direction. <Need to explain more>

Notes

None.

Exceptions

None.

Example

Example 1: Adding a clock exception from the pin `reg_q_a_10_/U0:CLK` with respect to the clock `clk`.

```
timer_add_clock_exception -clock {clk} -pin {reg_q_a_10_/U0:CLK}  
-dir {from}
```

Example 2: Adding a clock exception to the pin `reg_q_a_10_/U0:E` with respect to the clock `clk`.

```
timer_add_clock_exception -clock {clk} -pin {reg_q_a_10_/U0:E} -  
dir {to}
```

timer_add_pass

The `timer_add_pass` command adds the pin to the list of pins for which the path must be shown passing through, in the timer.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_add_pass  
  -pin pin_name
```

Arguments

`-pin pin_name`
Specifies the name of the pin to be included for displaying the timing path through it.

Notes

Setting a pass on a module pin enables you to see a path through individual pins.

Example

Example 1: Adding pass through the pin `reg_q_a_0_:CLK`

```
timer_add_pass -pin {reg_q_a_0_:CLK}
```

Example 2: Adding pass through a clear pin `reg_q_a_0_:CLR` in the design

```
timer_add_pass -pin {reg_q_a_0_:CLR}
```

See Also

`timer_add_stop`

timer_add_stop

The `timer_add_stop` command adds the specified pin to the list of pins through which the paths will not be displayed in the timer.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_add_stop -pin pin_name
```

Arguments

`-pin pin_name`
specifies the name of the pin to be excluded from displaying the path.

Notes

Without stop points, you see all the paths from pad to pad in the design. If you do not want to see the paths going through registers clock pins, you could specify these as stop points. The path going through those pins would not be displayed.

Exceptions

None.

Example

Example 1: Adding stop to the pin a<2>

```
timer_add_stop -pin {a<2>}
```

Example 2: Adding a stop to a clock and a clear pin in the design

```
timer_add_stop -pin {reg_q_a_0_:CLK}
```

```
timer_add_stop -pin {reg_q_a_0_:CLR}
```

See Also

Timer_add_pass

timer_commit

The `timer_commit` command saves the changes made to constraints into the Designer database.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_commit
```

Arguments

None.

Notes

None.

Exceptions

None.

Example

Example 1:
`timer_commit`

timer_get_path

The `timer_get_path` command obtains the path information from the timer and reports it to the log window.

Supported Family and Format

Family: All
Format: Tcl

Syntax:

```
timer_get_path -from source_pin -to destination_pin  
  ?-exp no | yes ?  
  ?-sort actual | slack?  
  ?-order long | short ?  
  ?-case worst | typ | best ?  
  ?-maxpath maximum_paths ?  
  ?-maxexpath maximum_paths_to_expand?  
  ?-mindelay minimum_delay?  
  ?-maxdelay maximum_delay?  
  ?-breakatclk no | yes ?  
  ?-breakatclr yes | no ?
```

Arguments

`-from source_pin`
specifies the source pin of the path.

`-to destination_pin`
specifies the destination pin for the path.

`?-exp no | yes ?`
specifies if the path needs to be expanded.

`?-sort actual | slack?`
specifies if the paths need to be sorted based on actual delay or the slack value.

`?-order long | short ?`
specifies if the maximum list size is based on longest or shortest paths.

`?-case worst | typ | best ?`
specifies if the report must consider timing values for the worst, typical or best cases.

`?-maxpath maximum_paths ?`
specifies the maximum number of paths to be reported.

`?-maxexpath maximum_paths_to_expand?`
specifies if number of maximum paths to be expanded.

`?-breakatclk no | yes ?`
specifies if the paths must be broken at the register clock pins

`?-breakatclr yes | no ?`
specifies if the paths must be broken at the register clear pins

Notes

None.

Exceptions

None.

Example

Example1:

```
timer_get_path -from "headdr_dat<31>" \  
-to "u0_headdr_data1_reg/data_out_t_31/U0:D" \  
-case typ \  
-exp "yes" \  
-maxpath "100" \  
-maxexpapth "10"
```

timer_get_clock_actuals

The `timer_get_clock_actuals` finds and reports the actual clock frequency when timer is initiated.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_get_clock_actuals -clock "name"
```

Arguments

List supported arguments here with an explanation of each argument.

Notes

The actual clock frequency is reported in the log window.

Exceptions

None.

Example

Example 1: Reports the actual clock frequency on the clock `clk1`

```
timer_get_clock_actuals -clock clk1
```

timer_get_clock_constraints

The timer_get_clock_constraints? <No idea what this does. No info anywhere>

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_get_clock_constraints -clock clock_name
```

Arguments

-clock *clock_name*
specifies the clock for which the constraint needs to be reported.

Notes

None

Exceptions

None.

Example

Example 1: Reports the clock constraint on the clock clk1

```
timer_get_clock_constraints -clock clk
```

timer_get_maxdelay

The `timer_get_maxdelay` command obtains maximum delay constraint between 2 pins of a path.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_get_maxdelay -from source_pin -to destination_pin
```

Arguments

`-from source_pin`
Specified the source pin of the path.

`-to destination_pin`
Specifies the destination pin of the path.

Notes

You can use the following macros in the command:

`$in()`
to specify all inputs.

`$out()`
to specify all output pins.

`$reg(clock_name)`
to specify all registers relates to *clock_name*..

Exceptions

None.

Example

Example 1: Get the max delay constraint from all registers of clk166 to all outputs

```
timer_get_maxdelay -from {$reg(clk166)[*]} -to {$out()[*]}
```

Example 2: Get the max delay constraint from the pin clk166 to the pin reg_q_a_9/U0:CLK

```
timer_get_maxdelay -from {clk166} -to {reg_q_a_9/U0:CLK}
```

Example 2: Get max delay constraint from all inputs to all registers of clock166 and also check for errors in the command.

```
if [ catch {timer_get_maxdelay -from {$in()[*]} -to  
{$reg(clk)[*]}} ] {
```

```
    puts "Error getting max_delay information"  
  } else {  
    puts "Successfully obtained max_delay information"  
  }  
}
```

See Also

`timer_set_maxdelay`

timer_get_path_constraints

The `timer_get_path_constraints` reports the path constraints that were set as max delay in the timer.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_get_path_constraints
```

Arguments

None.

Notes

If no max delay constraints are set this command will not report any delay values. <Can report "No valid path constraints found">. The information is displayed in the log window.

Exceptions

None.

Example

```
Timer_get_path_constraints.
```

timer_remove_clock_exception

The `timer_remove_clock_exception` command removes the previously set clock constraint.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_remove_clock_exception -clock clock_name -pin pin_name -dir  
{from}|{to}
```

Arguments

`-clock clock_name`

Specifies the clock name.

`-pin pin_name`

Specifies the exception to be removed on the pin in a synchronous network.

`-dir {from}|{to}`

Specifies the direction <Need to explain more>

Notes

None.

Exceptions

None.

Example

Example 1: Removing a clock exception from the pin `reg_q_a_10_/U0:CLK` with respect to the clock `clk`.

```
timer_remove_clock_exception -clock {clk} -pin {reg_q_a_10_/U0:CLK} -  
dir {from}
```

Example 2: Removing a clock exception to the pin `reg_q_a_10_/U0:E` with respect to the clock `clk`.

```
timer_remove_clock_exception -clock {clk} -pin {reg_q_a_10_/U0:E} -  
dir {to}.
```

See Also

`timer_add_clock_exception`

timer_remove_pass

The `timer_remove_pass` command removes the path pass constraint that has been previously entered.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_remove_pass -pin pin_name
```

Arguments

`-pin pin_name`
specifies the pin for which the path pass constraint must be removed.

Notes

None.

Exceptions

None.

Example

Example1 : removes the pass constraint from the clock pin `reg_q_a_0:CLK`.

```
timer_remove_pass -pin {reg_q_a_0:CLK}
```

See Also

`timer_add_pass`

timer_remove_stop

The `timer_remove_stop` command removes the path stop constraint on the specified pin that has been previously entered.

Supported Family and Format

Family: All
Format: TCL

Syntax

```
timer_remove_stop -pin pin_name
```

Arguments

-pin *pin_name*, specifies the pin for which the path stop constraint must be removed..

Notes

- Export of script writes the constraint wrong with `timer_remove_pass` instead of `timer_remove_stop`

Exceptions

None.

Example

Example 1: Removes the path stop constraint on the clear pin `reg_q_a_0_:CLR`.
`timer_remove_stop -pin {reg_q_a_0_:CLR}`

See Also

`Timer_add_stop`

timer_restore

The `timer_restore` command restores previous committed constraints.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_restore
```

Arguments

None.

Exceptions

None.

timer_setenv_clock_freq

The `timer_setenv_clock_freq` command sets a required clock frequency for the specified clock, in MHz.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_setenv_clock_freq -clock clock_name -freq frequency_value
```

Arguments

`-clock clock_name`
Specifies the clock for which the constraint is intended.

`-freq frequency_value`
Specifies the frequency in MHz.

Notes

None.

Exceptions

None.

Example

Example 1: Sets a clock frequency of 100MHz on the clock `clk1`

```
timer_setenv_clock_freq -clock {clk1} -freq 100.00
```

timer_setenv_clock_period

The `timer_setenv_clock_period` command sets the clock period constraint on the specified clock.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_setenv_clock_period -clock clock_name ?-unit {ns}|{ps}? -period  
period_value
```

Arguments

`-clock clock_name`

specifies the clock name for which the constraints in intended.

`?-unit "ns"|"ps"?`

optional argument specifies the unit for the clock period constraint. Default is "ns"

`-period period_value`

specifies the period in the specified unit.

Notes

None.

Exceptions

None.

Example

Example 1: Sets a clock period of 2.40ns on the clock clk1

```
timer_setenv_clock_period -clock {clk1} -unit {ns} -period 2.40
```

timer_set_maxdelay

The `timer_set_maxdelay` adds a maximum delay constraint for the path.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_set_maxdelay -from source_pin -to destination_pin ?-unit  
{ns}|{ps}? -delay delay_value
```

Arguments

- from *source_pin*
Specifies the source pin of the path.

- to *destination_pin*
Specifies the destination pin of the path.

- ?-unit "ns"|"ps"?
Specifies if the delay unit.

- delay *delay_value*
Specifies the actual delay value for the path.

Notes

You can use the following macros in the command:
\$in(), to specify all inputs.
\$out(), to specify all output pins.
\$reg(*clock_name*), to specify all registers relates to
clock_name.

Exceptions

None.

Example

Example 1: Set 20ns delay from all registers of clk166 to all outputs
`timer_set_maxdelay -from {$reg(clk166)*} -to {$out()*} -unit {ns} -delay 20.00`

Example 2: Set delay value of 30ns from all inputs to all outputs
`timer_set_maxdelay -from {$in()*} -to {$out()*} -unit {ns} -delay 30.00`

timer_remove_all_constraints

The `timer_remove_all_constraints` command removes all the timing constraints that have already been entered in the Designer system.

Supported Family and Format

Family: All
Format: Tcl

Syntax

```
timer_remove_all_constraints
```

Arguments

None.

Notes

None.

Exceptions

None.

Example

Example 1: Remove all constraints and commit the changes.

```
timer_remove_all_constraints  
timer_commit
```

See Also

`timer_commit`

SmartPower

Welcome to SmartPower

Welcome to SmartPower, the Actel power analysis tool. SmartPower supports only Flash and Axcelerator devices at this time; if you are not using a Flash or Axcelerator device, the SmartPower button disappears from your toolbar. For information on future support for other device families, visit the Actel website at <http://www.actel.com>.

When you launch SmartPower, you must first input your target clock and data frequencies before you evaluate your power consumption. There is no way to accurately measure the power consumption of your design without first entering your target clock and data frequencies.

For a complete description of the SmartPower tool, please refer to the *SmartPower User's Guide*, included with your user documentation.

Invoking SmartPower

You can only use SmartPower after you open a compiled design (*.adb file), or after compiling and layout of your netlist in Designer. If you invoke SmartPower before compiling your netlist, Designer guides you through the compile and layout.

There are three ways to invoke the SmartPower analysis tool:

1. Choose SmartPower from the Tools menu
2. Click the SmartPower icon in the Designer toolbar
3. Click the SmartPower button in Designer design flow

When you launch SmartPower for the first time, all clocks are assigned a frequency of 10 MHz by default. In addition, SmartPower sets all data frequencies to 1 MHz. In order for you to accurately measure the power consumption of a design, you must specify the target clock and data frequencies.

Steps to Calculate Power

Use the steps below to calculate your power consumption. The steps are identified by the tabs you must view (in the order you must view them) so that you may analyze your power accurately. Click the various tab names for a description of each tab. See the Power Analysis section of the online help for a complete explanation of each step.

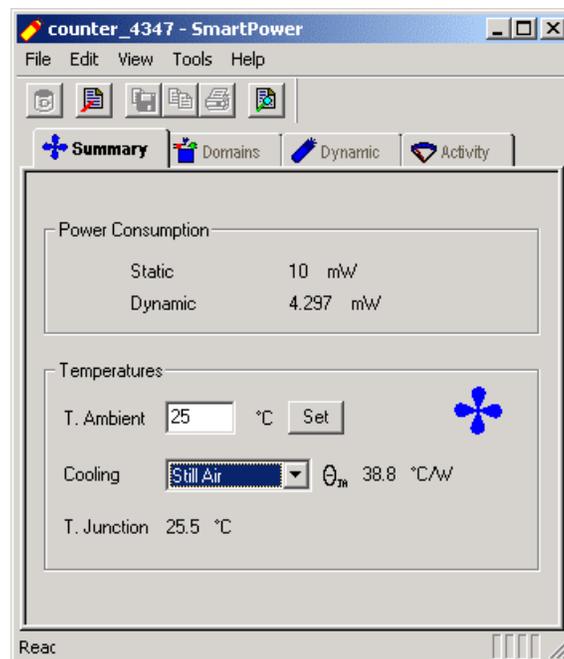
1. Domains tab - Define clock domains, and specify a clock frequency and a data frequency for each clock domain.

2. Activity tab - Advanced specification of frequencies. This step is optional, but gives you a pin-by-pin control of the frequency.
3. Summary tab - View global power at the design level and view its impact on Junction temperature.
4. Dynamic tab - View detailed hierarchical analysis of your power consumption. This step is also optional. But if your power consumption exceeds your budget, this step will help you to understand where there is room for improvement.

SmartPower Interface

Summary tab

The Summary tab is divided into two sections: Power Consumption and Temperatures.



SmartPower Summary Tab

Power Consumption

Displays the total static and dynamic power of the design. (Accurate only after you have entered your target clock and data frequencies!)

Temperatures

Displays the impact of the power consumption on the junction temperature for a given cooling scenario. You can specify a cooling scenario using the drop-down menu (available scenarios are: Still Air, 300 ft/min, Custom, and Case Cooling; default is still air). SmartPower also reports the thermal resistance, θ_{JA} .

The junction temperature estimation T_J is dependent on the thermal resistance (θ_{JA}) (which is itself package and cooling-style dependent), and also on the ambient temperature T_A and the total dynamic power consumption of your design P . The formula is:

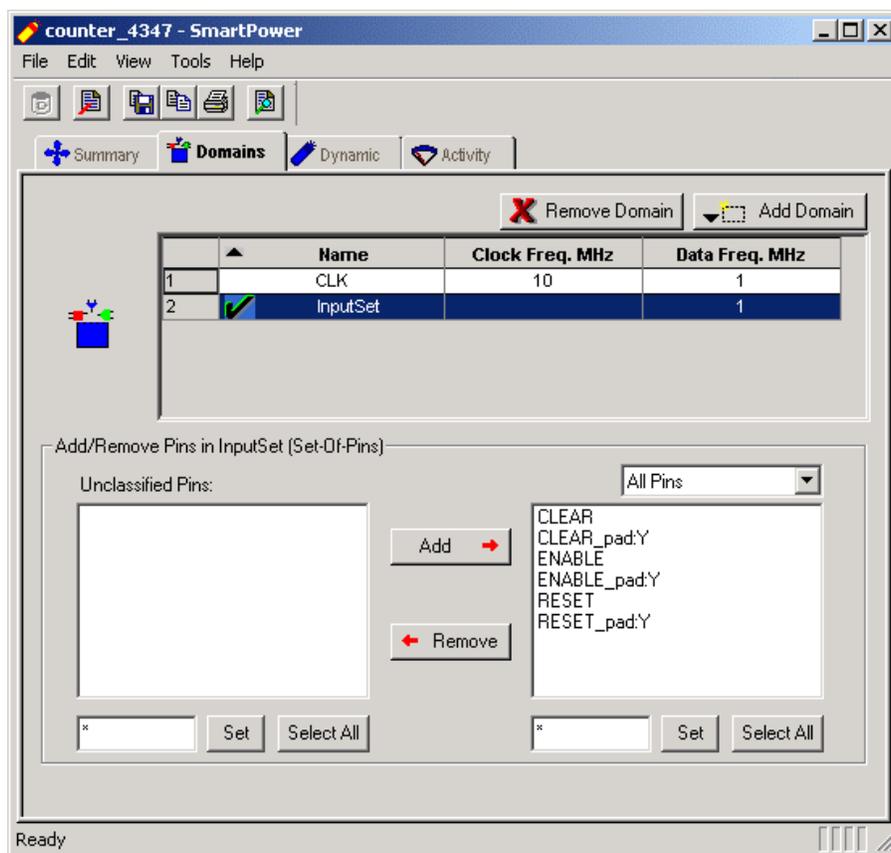
$$T_J = T_A + P \cdot \theta_{JA}$$

where:

$$\theta_{JA} = f(\text{Pkg \& Cooling Style})$$

Domains tab

The Domains tab consists of two windows: the Domain Management window and the Pin Management window. You can use these windows to add or remove domains. In addition, you can change the clock and/or data frequency of a selected domain.



SmartPower Domains Tab

Domain Management

The Domain Management window displays a list of existing domains with their corresponding frequencies.

The Domain Management window enables you to create new CLK domains or Set-Of-Pins. Also, You can delete or modify existing CLK or Set-Of-Pins domains.

To create a new CLK domain or Set-Of-Pins, click **Add Domain** and choose to add a new **Clock Domain** or **Set-Of-Pins**. Input the relevant information (potential clock pin, clock and data frequency for a clock domain; name and frequency for a pin) and click **Create**.

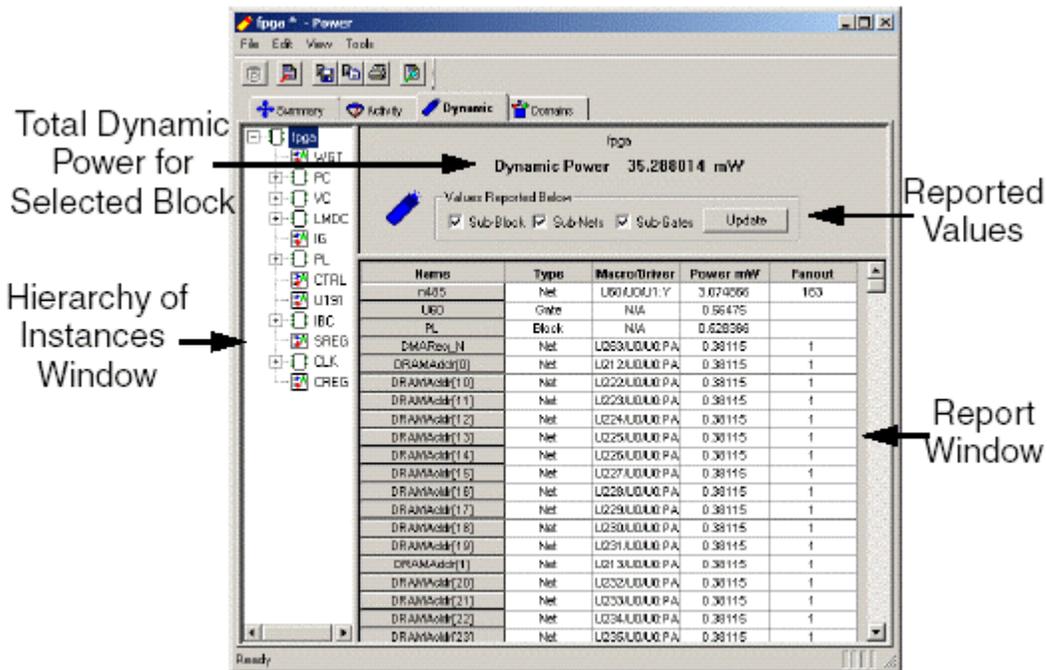
Pin Management

Any pins that do not belong to a domain are listed in the **Unclassified Pins** list box in the Pin Management window. You can select a pin from the **Unclassified Pins** list box and add it to the current domain. You may also select a pin from the current domain and remove it from the domain (this pin will appear in the **Unclassified Pins** list box).

Use the filter boxes to narrow your search for a specific pin. The boxes are text filters, * is a wildcard.

Dynamic tab

The Dynamic tab enables you to inspect detailed hierarchical reports of the dynamic power consumption. The Dynamic tab consists of two windows: the hierarchy of instances window, and the report window.



SmartPower Dynamic Tab

Hierarchy of Instances Window

SmartPower displays the hierarchy of instances in a list in the hierarchy window. Sub-blocks of a block are shown in the tree when you click the plus sign (+) next to the block. Only hierarchical blocks are displayed in this list (no gates or nets).

When you select a block of the hierarchical tree, SmartPower displays its name and its dynamic power consumption in the report window.

Report Window

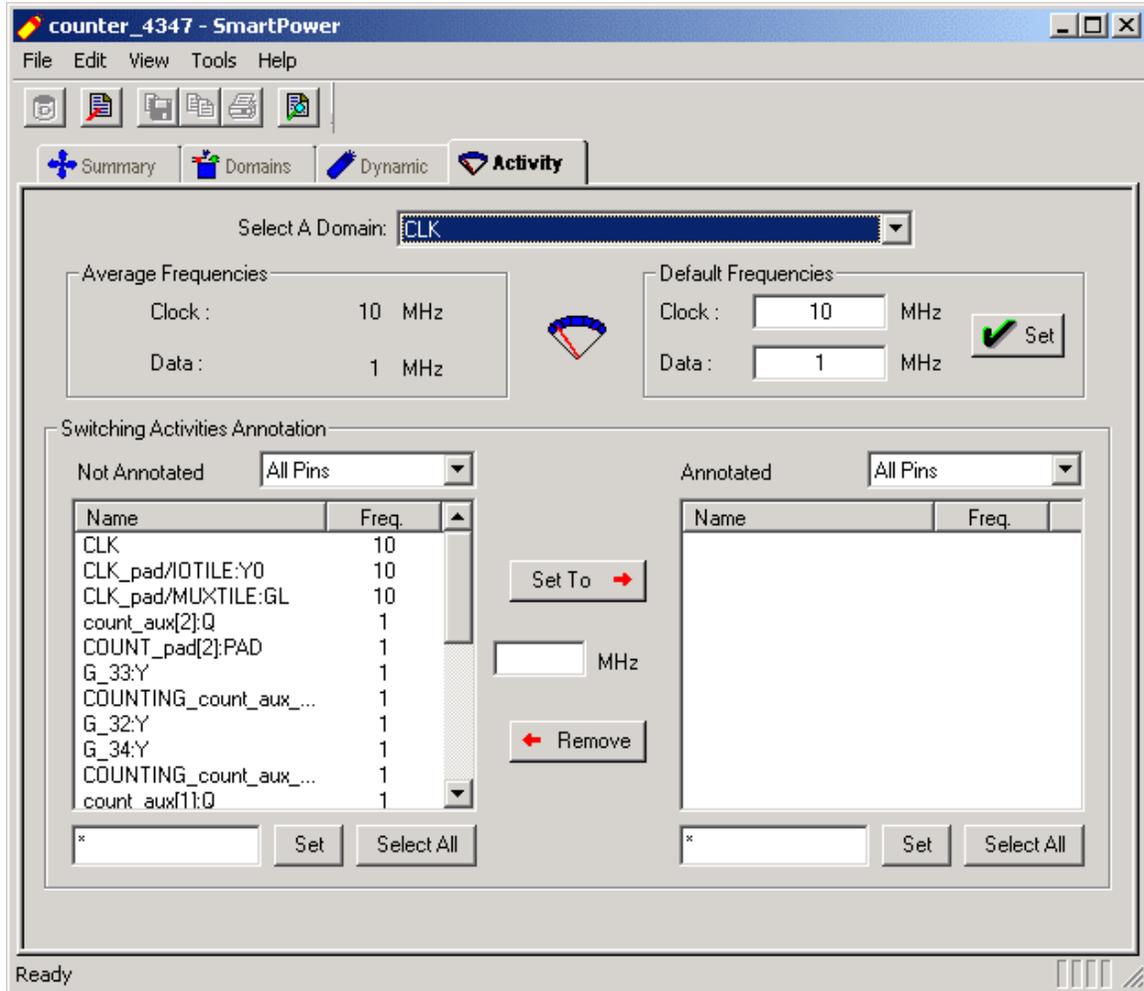
SmartPower displays the list of sub-elements of the selected block in the Report window. By default, this list includes all sub-elements. The dynamic power consumption of each sub-element is displayed with useful information like the fanout and the driver-name for a net, or the macro model-name for a gate.

You may limit the list of sub-elements to a list of sub-blocks, or gates, or nets, or any combination of these 3 classes of sub-elements. You may also sort the list according to different criteria (double-click a column label to sort the list based on this column, or change the sort-order).

You can export (to a text file) and print the grid that details your design's power consumption. To do so, select the elements of the grid that you wish to export or print, and then from the File menu select Export Grid or Print Grid, respectively.

Activity tab

Use the Activity tab to attach switching activity information on interconnects of the design. The Activity tab is divided into the Select A Domain drop-down menu, the Frequency Estimation area, and the Switching Activities Annotation area.



SmartPower Activity Tab

Select a Clock Domain Drop Down Menu

Specifies the clock domain (or set of pins). Use the drop-down menu to select a different domain.

You can create your own unique set of pins in the Domains tab.

Average Frequencies

Includes the average frequency of the clock-pins and data-pins of the selected clock domain. Use the Select a Domain drop-down menu to choose another clock domain. If you wish, you may select a set of pins rather than a clock domain. If you select a set of pins instead of a clock domain, SmartPower reports only one average frequency (the average frequency of all the pins of the selected set).

Average Frequencies are useful when you import a VCD file or SAIF file. Since these files enable you to specify the frequency of each pin individually, it is often useful to know the average clock-pin or data-pin frequency for a particular clock domain.

To view the Average Frequencies of a clock domain:

1. Select a clock domain. Click the **Activity** tab, and select a specific domain in the list.
2. **Verify the average clock frequency.** If you did not specify a frequency annotation for any clock-pin in this clock domain, the average value is equal to the default clock-frequency of the clock domain. If you annotated one or several clock-pins, SmartPower takes these specific annotations into account to compute an average value.
3. **Verify the average data frequency.** If you did not specify a frequency annotation for any data-pin in this clock domain, the average value is equal to the default data-frequency of the clock domain. If you annotated one or several data-pins, SmartPower takes these specific annotations into account to compute an average value.

Default Frequencies

Enables you to specify the global clock frequency and data frequency for the given clock domain (or set of pins). For designs with multiple clocks, SmartPower defaults to the first clock in alphabetical order. If you wish, you may select a set-of-pins rather than a clock-domain. In this case, you can modify only one frequency (this frequency is used for all the pins of the selected set).

Switching Activities Annotation Area

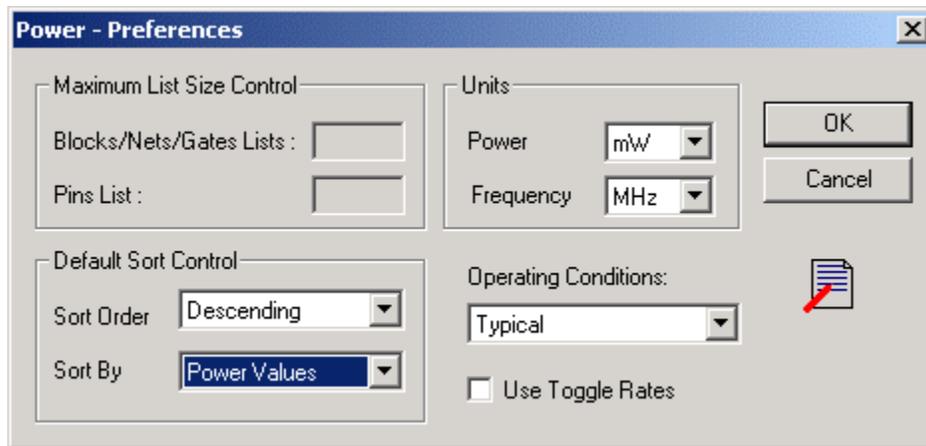
Enables you to specify the switching activities for individual pins in the **Clock Domain**. SmartPower displays the pins that have not been annotated in the Not Annotated list box.

Select a pin and specify a different frequency for this pin using the text-box and the **Set To** button. When you select a pin and specify a frequency, SmartPower removes the pin from the **Not-Annotated** list-box and adds it to the **Annotated** list-box. Hold down the CTRL key and click with the mouse to select multiple pins.

Use the **Select All** button to select all the pins in a list-box. Filter boxes are provided below the list boxes to limit the size of each list of pins. Enter text in these boxes and click **Set** to apply this text as a filter (the * character is a wildcard). It is also possible to limit the type of each list of pins using a drop-down menu that enables you select **All-Pins**, **Data-Pins** or **Clock-Pins**.

SmartPower Preferences

Enables you to set options that affect the graphical and textual reports. To open the **SmartPower - Preferences** dialog box, from the **File** menu choose **Preferences**. Alternatively, you can click **Options** in the **Power Report** dialog box.



SmartPower Preferences Dialog Box

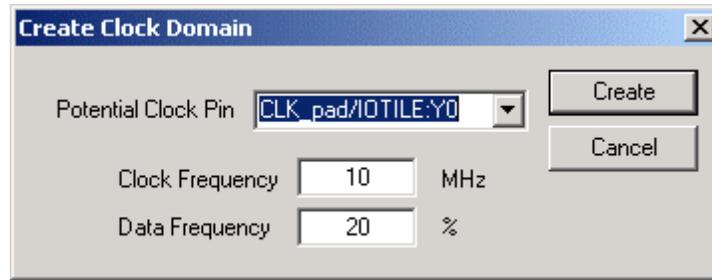
There are four sections: **Maximum List Size Control**, **Default Sort Control**, **Units**, and **Operating Conditions**.

- **Maximum List Size Control:** Enables you to limit the size of all lists displayed in the SmartPower tab screens (option unavailable at this time).
- **Default Sort Control:** Modifies the default sort for all the lists in SmartPower (available sort keys are Alphabetical or Power values, in either ascending or descending order).
- **Units:** Sets unit preferences for power and frequency.
- **Operating Conditions:** Displays operating conditions; Typical is the only option available at this time.

Use Toggle Rates

When toggle rates are active (Toggle Rates box is checked), the data frequency of all the Clock Domains are defined as a function of the percentage of the clock frequency. This updates the data frequency automatically when you update the clock frequency. Toggle Rates enable you to specify the data frequency as a percentage of clock frequency, but you can no longer specify the actual data frequency, only a percentage value.

Set the data frequency percentage when you create a new clock domain with Toggle Rates active. Also, when toggle rates are active you can set the data frequency percentage in the Domain and Activity tabs.



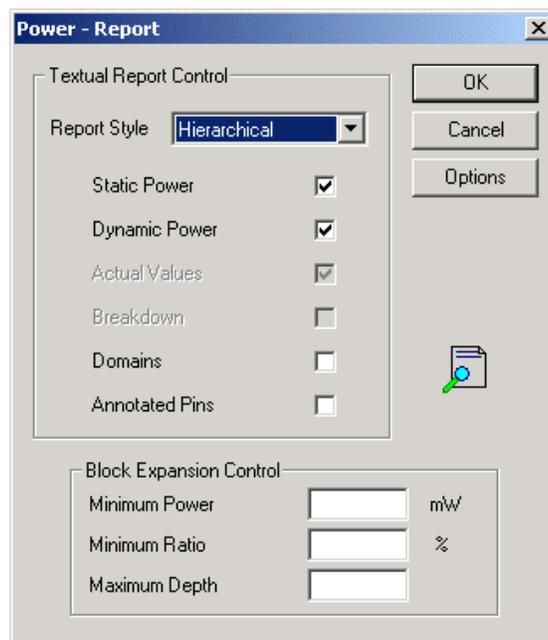
Create Clock Domain - Toggle Rates Enabled

Power Reports

The power report enables you to quickly determine if any power consumption problems exist in your design. The power report lists the following information:

1. Global device information and SmartPower Preferences selection information
2. Dynamic power summary
3. Design-level static power summary
4. Hierarchical detailed power report (including gates, blocks, and nets), with a block by block, gate by gate, and net by net power summary

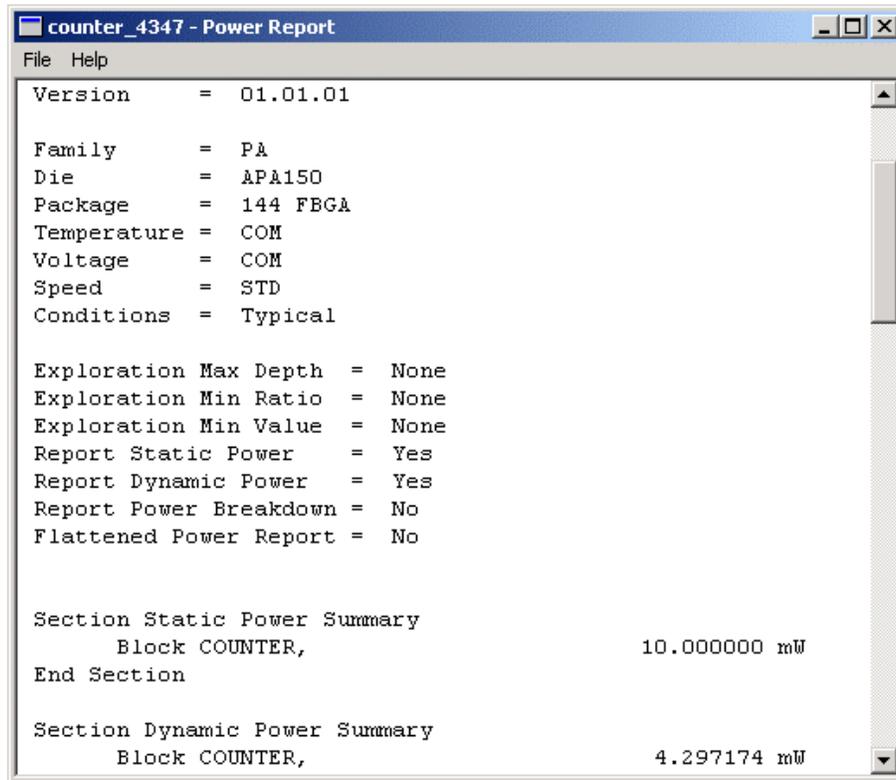
Click the Report button to open the Report dialog box. Specify which results you want to display (static or dynamic Power).



SmartPower Report Dialog Box

The SmartPower report returns a complete list of all the blocks, gates, and nets and the related power consumption in the device (it returns the same information displayed in the **Dynamic** tab, but it is more printer friendly).

Set the options in the **Textual Report Control** to customize your power report. Click the checkboxes to include information on **Static Power**, **Dynamic Power**, **Domains**, and **Annotated Pins** (Actual Values are included in all power reports; Breakdown information is not available at this time).



SmartPower Report

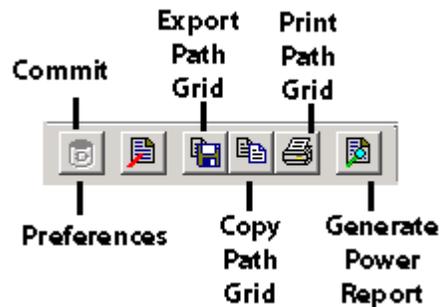
The report fully expands all the info included in the **Dynamic** tab by default; use the **Block Expansion Control** to expand only the blocks you are interested in.

The **Block Expansion Control** options filter the values returned in the report. Block Expansion does not control which values are included (the Textual Report Control options determine content), rather it specifies which blocks are detailed or expanded.

You may specify which blocks are expanded using a **Minimum Power** value, a **Minimum Ratio** (with regards to the total power of the design) and a **Maximum** (hierarchical) **Depth**; a value filtered by Block Expansion Control is not included in displayed lists, but it is still included in the upper hierarchical analysis of the design. In other words, SmartPower still includes filtered values in the power analysis.

SmartPower toolbar / menu commands

The SmartPower toolbar contains commands for performing common SmartPower operations on your designs. Roll the mouse pointer over the toolbar button to view a description of the button. Click the button to access the command.



SmartPower Toolbar

The PC and workstation versions of SmartPower have the same menus. However, some dialog boxes may look slightly different on the two platforms due to the different window environments. The functionality is the same on both platforms, though the locations of the fields and buttons on the dialog boxes may vary. The names of some fields may also vary between the PC and workstation versions.

File Menu

Commit: Commits power information to Designer. You must commit your changes if you wish to save your settings in SmartPower. If you commit your changes, the information is stored in the .adb file, and your settings are restored the next time you open your design in SmartPower.

Export Grid (enabled in Dynamic tab): Exports the selected area of the Report Window to a text (.txt) file.

Print Grid (enabled in Dynamic tab): Prints the selected area of the Report Window.

Preferences: Invokes Preferences dialog box, where you can set analysis and display preferences

Close: Closes SmartPower

Edit Menu

Add Domain (enabled in the Domains tab): Adds a clock domain or set of pins.

Remove Domain (enabled in the Domains tab): Removes a domain.

Copy Grid: Copies the selected cells of the dynamic grid onto the clipboard

Tools Menu

Report Power: Generates power report

Importing a VCD file in Designer

The VCD (value change-dump) file is a file format specified in the IEEE 1364 standard. It is an ASCII file that contains header information, definition of variables, and the values of variables.

You can generate a pre- or post-layout VCD file using ModelSim or any other simulation tool that supports VCD file generation. Please refer to the user manual of your simulation tool for more information on how to generate a VCD file.

To import a VCD file:

1. From the **File** menu in **Designer**, select **Import Auxiliary Files**. Click **Add** to browse to your VCD file and select it. When you have selected a VCD file, click **OK** to continue.

If you have not yet completed the layout of the design, the design software guides you through place-and-route so that you can import the VCD file. In order to successfully annotate your VCD values to the design, Designer must complete place-and-route even if you generated your VCD file using timing simulation (post-layout).

You may wish to import multiple VCD files. If these files conflict (attempt to set a different frequency for the same net of your design, for example), the latest imported value takes precedence.

2. Specify your VCD import options. Use the **VCD Import Options** window to specify the instance name of your design in the simulation testbench (the instance name is the instance name of your design instantiated in the simulation testbench). For example, the instance name of the design “top_comp” in the following verilog test-bench is “inst”.

```
module test;  
  reg [3:0] DataA, DataB;  
  wire AGEB;  
  top_comp inst(DataA, DataB, AGEB);  
endmodule
```

```
initial
begin
.....
end;
endmodule;
```

It is also possible to identify the instance name of your design in the VCD file. You have to look for a line starting with the keyword `$scope`. For example, the instance name of the design “top_comp” in the following VCD file is “inst”.

```
$date
Oct 18, 2001 16:02:16
$end
$version
VERILOG-XL 3.30.p001
$end
$timescale
100ps
$end
$scope module inst $end
.....
```

Click **OK** to continue.

3. Check the **Log** window for notification that you successfully imported the VCD file (“The Import command succeeded...”). Even if the Import command succeeds, Actel recommends that you use SmartPower to verify which of the pins have been affected after you import the file.
4. Verify results of the imported file in the **Activity** tab screen in SmartPower. To view the results of your imported VCD file, launch SmartPower and navigate to the **Activity** tab screen to view pins with annotated switching activities. If your file was imported successfully, you will see a long list of pins with annotated switching activity and specific individual frequencies.

It may be that some pins of your design are not annotated by a VCD import command. This happens if you simulate a pre-synthesis netlist; it is normal because not all logic elements are in the pre-synthesis netlist. Thus, for accurate power estimation it is better to run post-layout simulation with a back-annotated netlist.

Importing a SAIF file in Designer

The following instructions are meant only as a guide. For an explanation of the complete flow (including screenshots), please refer to the SmartPower User's Guide included with your software.

To import a SAIF file:

1. From the **File** menu in **Designer**, select **Import Auxiliary Files**. Click **Add** to browse to your SAIF file and select it. When you have selected a SAIF file, click **OK** to continue.

If you have not yet completed the layout of the design, the design software guides you through place-and-route so that you can import the SAIF file. In order to successfully annotate your SAIF values to the design, Designer must complete place-and-route even if you generated your SAIF file using timing simulation (post-layout).

You may wish to import multiple SAIF files. If these files conflict (attempt to set a different frequency for the same net of your design, for example), the latest imported value takes precedence.

2. Specify your SAIF import options. Use the **SAIF Import Options** window to specify the instance name of your design in the simulation testbench (the instance name is the instance name of your design instantiated in the simulation testbench). **You must include the hierarchy with the instance name.**

The example below example shows how to identify the instance name of your design in the SAIF file. For example, the instance name of the design in the following SAIF file is "TEST_BENCH/UUT".

```
( SAIFILE
( SAIFVERSION "1.1" )
( DESIGN 2ff )
( DATE "Fri May 10 14:48:46 2002" )
.....
```

```

(TIMESCALE 1ns)
(DURATION 50000)
(INSTANCE TEST_BENCH/UUT (PORT (OUT_PORT (TC 26) (IG
0) (T1 25994)
(T0 22000) (TX 2006))))
(INSTANCE TEST_BENCH/UUT/\outpad/U0/U1\ (PORT (Y (TC
26) (IG 0)
(T1 25995) (T0 22000) (TX 2005))))
(INSTANCE TEST_BENCH/UUT/\ff1/U0\ (PORT (Q (TC 27) (IG
0) (T1 26000)
(T0 22997) (TX 1003))))
(INSTANCE TEST_BENCH/UUT/\clkpad/U0/U0\ (PORT (Y (TC
99) (IG 0)
(T1 25000) (T0 24999) (TX 1))))
.....

```

Click **OK** to continue.

3. Check the **Log** window for notification that you successfully imported the SAIF file (“The Import command succeeded...”). Even if the Import command succeeds, Actel recommends that you use SmartPower to verify which of the pins have been affected after you import the file.
4. Verify results of the imported file in the **Activity** tab screen in SmartPower. To view the results of your imported SAIF file, launch SmartPower and navigate to the **Activity** tab screen to view pins with annotated switching activities. If your file was imported successfully, you will see a list of pins with annotated switching activity and specific individual frequencies.

It may be that some pins of your design are not annotated by a SAIF import command. This sometimes happens if you simulate a pre-synthesis netlist. It is normal; not all logic elements are in the pre-synthesis netlist. Thus it is better to do a post-layout simulation with a back-annotated netlist for the most accurate power estimation.

Steps to calculate power

Use the steps below to calculate your power consumption. The steps are identified by the tabs you must view (in the order you must view them) so that you may analyze your power accurately.

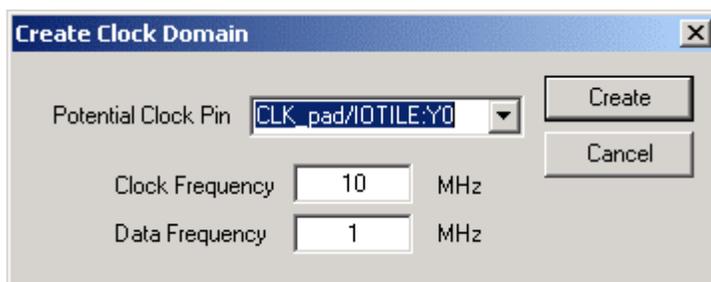
1. Domains tab - Define clock domains, and specify a clock frequency and a data frequency for each clock domain.
2. Activity tab - Specify individual pin frequencies, this step is optional, but gives you a pin-by-pin control of the frequency.
3. Summary tab - View global power at the design level and view its impact on Junction temperature.
4. Dynamic tab - View detailed hierarchical analysis of your power consumption. This step is also optional. But if your power consumption exceeds your budget, this step will help you to understand where there is room for improvement.

Define clock domains and Set-Of-Pins

When you run SmartPower, it researches your existing clock domains and partitions your design automatically. You may wish to review the list of clock domains in the Domains tab to ensure that all the clocks of your design are included in the list. Add or remove clocks as necessary.

To add a new clock domain:

1. Click the **Domains** tab, and click the **Add Domain** button. Select **Clock Domain** from the drop-down menu. This opens the **Create Clock Domain** dialog box.

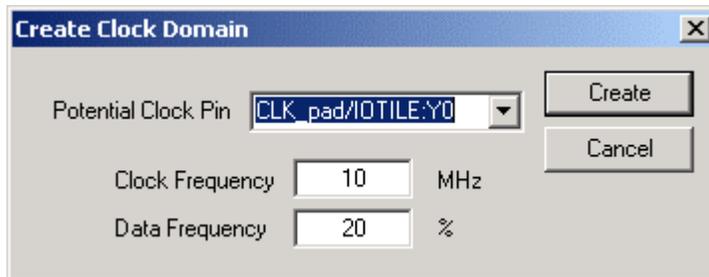


Create Clock Domain Dialog Box - Toggle Rates Disabled

2. To create a new clock, select a **Potential Clock Pin**, specify a clock and data frequency, and click **Create**. The new clock domain appears in the **Domains** window. If you select an existing clock-pin from the drop-down menu, the lists of

clock-pins and data-pins of this new clock domain are computed automatically based on the netlist topology.

Note: Select Use Toggle Rates in the SmartPower Preferences, to define your data frequency as a percentage of your clock frequency. If your data frequency is 20% of your clock frequency, type "20" in the Data Frequency text box.



Create Clock Domain Dialog Box - Toggle Rates Enabled

You may wish to create an empty clock domain and fill the lists of clock-pins and data-pins manually. If so, do not select a clock-pin, just type a new name for your clock domain.

Beyond the verification of the list of clock domains, you may also wish to verify that the lists of clock-pins and data-pins computed for each clock domain are correct.

To verify the lists of clock-pins and data-pins of a clock domain:

1. To select a Clock Domain, click the **Domains** tab, and select a specific **Domain** in the list.
2. **Display the list of clock-pins or data-pins of this Domain.** A drop-down menu in the **Domain** tab enables you to select clock-pins or data-pins. SmartPower displays the list of pins corresponding to your selection below the drop-down menu. You can add or remove clock-pins and data-pins as necessary.
3. **Remove a pin from a clock domain.** Highlight the selected pin and click the Remove button. The pin is removed from the clock domain, and is made available in the list of pins that you can add in another clock domain.
4. Highlight the selected pin in the list of pins that are not yet in a domain and click the Add button to add a pin in a clock domain. This pin is added to the clock domain. It is a clock-pin or a data-pin, depending on the specification of the

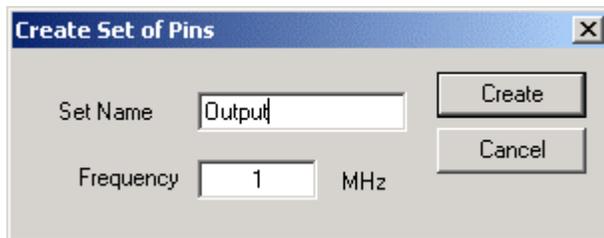
drop-down menu when you click the **Add** button.

Note: You cannot add a pin that exists in another domain until you free it from the existing domain. The pin is unavailable until you remove it from the existing domain.

After you have verified that all the clocks of your designs are correctly identified and constructed, you must specify the correct clock and data frequency for each clock domain.

To add a new set of pins:

1. Open the **Create Clock Domain** dialog box. Click the **Domains** tab, and click the **Add Domain** button. Select **Set-Of-Pins** from the drop-down menu.



Create Set of Pins Dialog Box

2. **Create a Set-Of-Pins.** Name your new Set-Of-Pins, specify a data frequency, and click **Create**. The new Set-Of-Pins appears in the **Domains** window.

Specify clock and data frequencies in SmartPower

To specify a clock and data frequency, highlight the **Clock/Data** frequency cell and type in a new value.

SmartPower defaults to 10 MHz for each clock frequency, and 1MHz for the data frequency. Input your target for each clock and data frequency (5% of your clock frequency is a typical guideline for your data frequency - this corresponds to a toggle-rate of 10%.)

Not all the pins/gates/nets of your design are associated with a specific Clock. For example, the frequency of a design input port is not always correlated to a clock frequency. By extension, all pins that are upstream of the first level of sequential elements are not associated with any clock. SmartPower creates an InputSet by default that it uses to group all the pins that are controlled by design inputs (instead of sequential elements). You may wish to view and verify the InputSet to further evaluate your design.

View and verify the InputSet in SmartPower

To verify the InputSet:

1. Select the **InputSet**. Click the **Domains** tab, and select the domain named **InputSet** in the list.
2. Verify the list of pins of this Domain. All the input ports of your design (except the clocks) belong in the **InputSet**. Also, all the pins that are between these input ports and the first level of sequential elements belong in the **InputSet**. You can add or remove pins as necessary.
3. Specify an average input frequency. SmartPower uses the same frequency for all pins of the InputSet. The default InputSet frequency is 1 MHz. Type in a new value to change it.

You may wish to split the InputSet into several sets in order to specify different frequencies. A classic example is to create a ResetSet, a reset-tree with a very low frequency.

To split the InputSet into several sets:

1. Create a new Set of Pins. In the Domains tab, click the New button, and select Set-of-Pins from the drop down menu. In the Create Set Of Pins dialog-box type a name and a frequency for the new set and click Create. The new set of pins appears in the Domains window. You can only create an empty set of pins, but it is possible to add pins in this Domain latter.
2. Remove a group of pins from the InputSet. Click the Domains tab, and select the domain named InputSet in the list. Highlight the pins that you want to remove and click the Remove button.

3. Add this group of pins in the new Set of Pins. Click the Domains tab, and select the newly created set of pins in the list. Highlight the pins in the list of pins that are not yet in a domain, and click the Add button. Repeat these 3 steps as necessary to create multiple inputs sets.

Specify individual pin frequencies

The Activity enables you to specify an average clock and data frequency for each clock domain, and also an average frequency for each set of pins. This gives you an initial estimate of the power consumption of your design. However, if this estimate is not accurate enough, you may refine it with a pin-by-pin annotation of the frequency.

To specify a frequency annotation for an individual pin:

1. Locate the pin in the **Activity tab**. You may need to select different clock domains from the drop-down menu on the **Activity tab**, then search in the **Not-Annotated Pins list** to find the specific pin. You can use filters to facilitate this search.
2. Highlight the pin in the list of **Not Annotated** pins, enter a new frequency value, and click the **Set To** button. This specifies a new frequency for the selected pin. The pin with this new frequency appears in the list of **Annotated** pins. Repeat these 2 steps as necessary to annotate the frequency of several pins.

Note: This annotation procedure enables you to set the frequency of an individual pin, but this does not mean that the pin is removed from its clock-domain. A frequency annotation just overrides the domain-level frequency.

You may wish to change or remove a frequency annotation of an individual pin. This may be useful when you import a VCD (value change-dump) file or a SAIF (Switching Activity Interchange Format) file.

To change the frequency annotation of an individual pin:

1. Locate the pin in the **Activity** tab. You may need to select different clock domains from the drop-down menu on the **Activity** tab, and then search in the **Annotated Pins list** to find the specific pin. You can use filters to facilitate the search.
2. Highlight the pin in the list of **Annotated** pins, enter a new frequency value, and click the **Set To** button. This specifies a new frequency for this pin. The pin appears in the list of annotated pins with this new frequency. Repeat these 2 steps as necessary to change the frequency annotation of several pins.

To remove the frequency annotation of an individual pin:

1. Locate the pin in the **Activity** tab. You may need to select different clock domains from the drop-down menu on the **Activity** tab, and then search in the **Annotated Pins list** to find the specific pin. You can use filters to facilitate the search.
2. Highlight the pin in the list of **Annotated** pins and click the **Remove** button. This removes the specified frequency from the Annotated pin. The pin appears in the list of **Not Annotated** pins. Repeat these 2 steps as necessary to remove the frequency annotation of several pins.

View results (design level)

Click the **Summary** tab to view global power consumption at the design level. The **Summary** tab shows your designs' estimated **Power Consumption** and **Temperature** information.

The power estimation reported in the **Summary** tab is the total static and dynamic power consumption of your design. For a more detailed view of this power consumption, click the **Dynamic** tab.

To estimate the junction temperature:

1. Verify your package. You cannot change your package directly in SmartPower, because it may obsolete your place-and-route information (and thus it may severely impact the total power consumption). If you wish to choose another package, you have to do it in **Designer -> Tools -> Device Selection**.
2. Click the **Summary** tab, and select a **Cooling** style in the list. Thermal resistance changes automatically when you update the cooling style.

- Specify an ambient temperature. Enter an **ambient temperature** (default value is 25°C), and click the **Set** button.

Note: The junction temperature value changes according to the package, cooling style, and ambient temperature values you choose.

Analyze results

The **Dynamic** tab displays the estimated power consumption of individual blocks, gates, and nets and enables you to make a hierarchical analysis of your power consumption. The **Dynamic** tab may also help you to improve your power consumption by identifying the blocks, gates and nets consuming a significant amount of power.

You can export (to a text file) and print the grid that lists your design's power consumption. To do so, select the elements of the grid that you wish to export or print, and then from the **File** menu select **Export Grid** or **Print Grid**, respectively.

To identify the blocks, gates, or nets that are consuming the most power:

- Use the **Dynamic** tab to expand the design hierarchy. The **Dynamic** tab enables you to expand your design hierarchy and view a complete list of the blocks in your design. Click the '+' next to your design to view the hierarchy. Click the '+' next to a sub-block to view its sub-elements. Consider the figure below, which shows an example of a clock pin with high fanout.

Block Level Hierarchy →

← Blocks, Nets and Gates

Name	Type	Macro/Driver	Power mW	Fanout
m005	Net	U000000000	0.074406	163
U00	Gate	N/A	0.004975	1
U0	Block	N/A	0.000000	1
DRAMAddr[10]	Net	U223A0U0PA	0.38115	1
DRAMAddr[11]	Net	U223A0U0PA	0.38115	1
DRAMAddr[12]	Net	U223A0U0PA	0.38115	1
DRAMAddr[13]	Net	U223A0U0PA	0.38115	1
DRAMAddr[14]	Net	U223A0U0PA	0.38115	1
DRAMAddr[15]	Net	U223A0U0PA	0.38115	1
DRAMAddr[16]	Net	U223A0U0PA	0.38115	1
DRAMAddr[17]	Net	U223A0U0PA	0.38115	1
DRAMAddr[18]	Net	U223A0U0PA	0.38115	1
DRAMAddr[19]	Net	U223A0U0PA	0.38115	1
DRAMAddr[20]	Net	U223A0U0PA	0.38115	1
DRAMAddr[21]	Net	U223A0U0PA	0.38115	1
DRAMAddr[22]	Net	U223A0U0PA	0.38115	1
DRAMAddr[23]	Net	U223A0U0PA	0.38115	1

2. **Click to select a block.** By default SmartPower selects the design-level block, but you can always select another block in the hierarchical tree. The report window displays the list of sub-elements of the selected block. By default, this list includes all sub-elements. SmartPower displays the dynamic power consumption of each sub-element with useful information like the fanout and the driver-name for a net, or the macro model-name for a gate.
3. **Sort and filter the sub-elements to find the block, gate, or net that is using the most power.** Double-click a column heading to sort by that column (or to change the sort order). By default SmartPower sorts the sub-elements according to their power consumption. The top of the list of sub-elements gives you the main sources of dynamic power consumption at the hierarchical level. Click a checkbox to limit the list of sub-elements to a list of gates, nets or blocks.

Cross probing with SmartPower

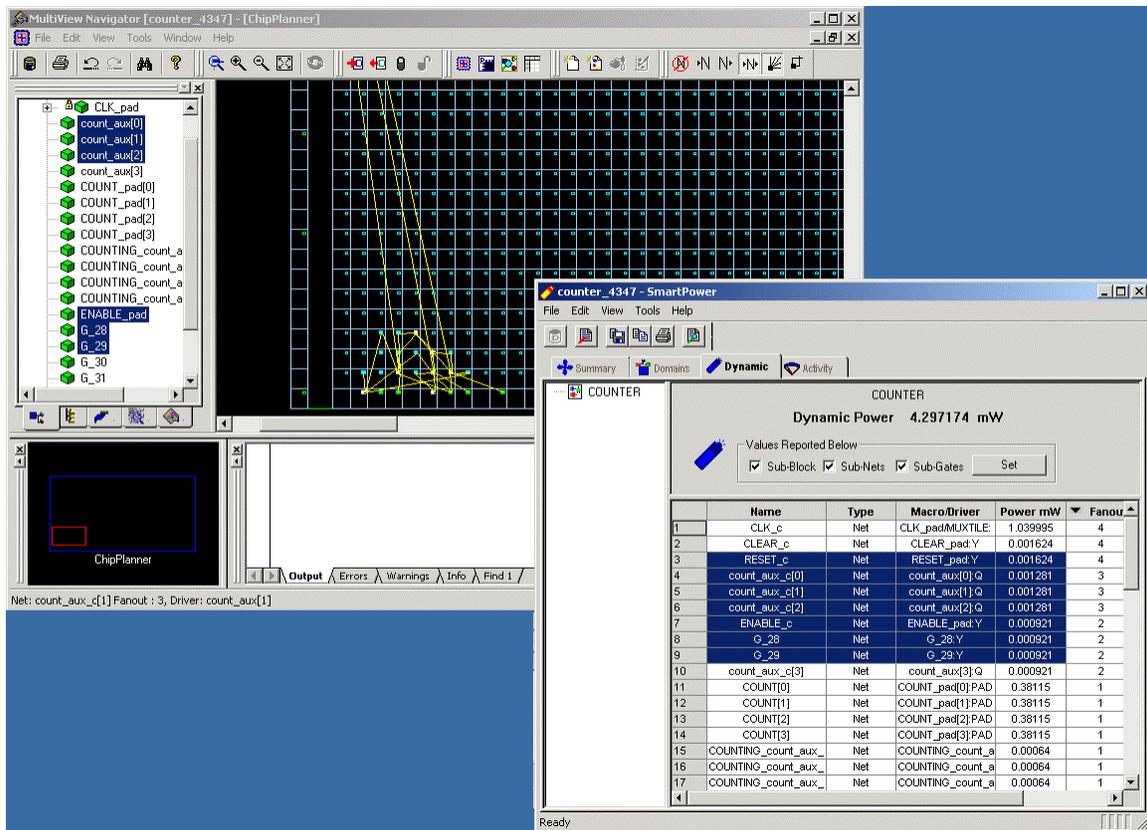
SmartPower supports cross probing with the other Designer tools. You must calculate your designs power consumption before you can cross probe effectively. See the Calculating Power section for more information.

To cross probe with the SmartPower tool:

1. View the detailed results of your power analysis in the **Dynamic** tab of the SmartPower tool.
2. Open the **ChipPlanner** or the **PinEditor** tool in Designer.
3. Click a block, net, or gate in the **Dynamic** tab to highlight the corresponding component in the **ChipPlanner** or **PinEditor** tool.

Click the **Macro/Driver** or **Name** column to cross probe gates.

Click an object in the **Name** column to select individual nets; click an object in the **Macro/Driver** column to select the object connected to the net.



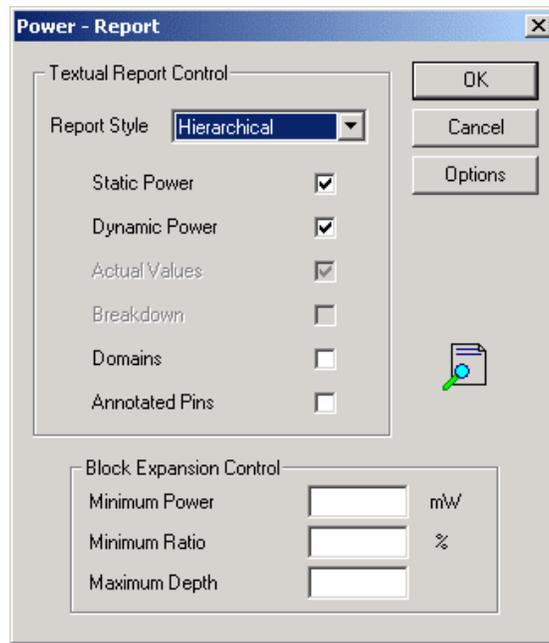
Cross Probing with SmartPower

Power Reports

The power report enables you to quickly determine if any power consumption problems exist in your design. The power report lists the following information:

1. Global device information and SmartPower Preferences selection information
2. Dynamic power summary
3. Design-level static power summary
4. Hierarchical detailed power report (including gates, blocks, and nets), with a block by block, gate by gate, and net by net power summary

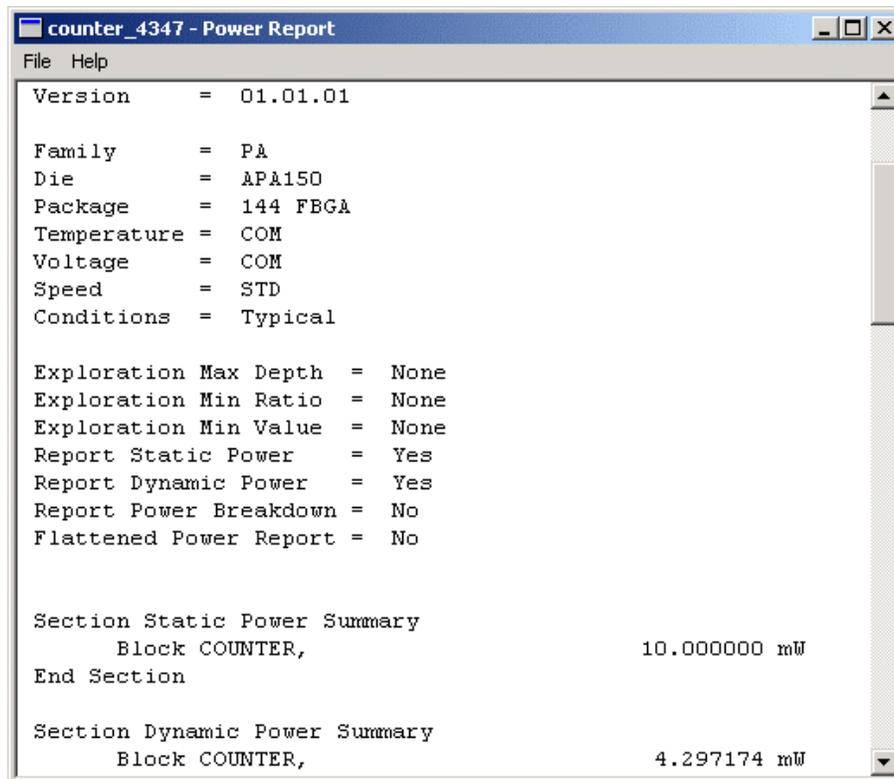
Click the Report button to open the Report dialog box. Specify which results you want to display (static or dynamic Power).



SmartPower Report Dialog Box

The SmartPower report returns a complete list of all the blocks, gates, and nets and the related power consumption in the device (it returns the same information displayed in the **Dynamic** tab, but it is more printer friendly).

Set the options in the **Textual Report Control** to customize your power report. Click the checkboxes to include information on **Static Power**, **Dynamic Power**, **Domains**, and **Annotated Pins** (Actual Values are included in all power reports; Breakdown information is not available at this time).



SmartPower Report

The report fully expands all the info included in the **Dynamic** tab by default; use the **Block Expansion Control** to expand only the blocks you are interested in.

The **Block Expansion Control** options filter the values returned in the report. Block Expansion does not control which values are included (the Textual Report Control options determine content), rather it specifies which blocks are detailed or expanded.

You may specify which blocks are expanded using a **Minimum Power** value, a **Minimum Ratio** (with regards to the total power of the design) and a **Maximum** (hierarchical) **Depth**; a value filtered by Block Expansion Control is not included in displayed lists, but it is still included in the upper hierarchical analysis of the design. In other words, SmartPower still includes filtered values in the power analysis.

Power Calculation Theory

SmartPower equations

SmartPower calculates two power values for your design:

1. Static Power - This value is family and die-size dependent and is estimated at the design level (at this time all die sizes for each family have the same static power).
2. Dynamic Power: This value is a summation of the dynamic power consumed by each element of the design (nets, modules, IOs, RAM, FIFO, PLL, etc.).

Note: The examples below are for general evaluation purposes only. They are not a precise representation of the actual calculations, since each calculation takes into account family-specific information.

- For a net,

$$P = C \cdot V^2 \cdot F$$

where C is the total capacitive loading of the net (extracted from the routing topology), V is the net's voltage swing, and F is the average switching frequency.

- For a module, the power is computed using a characterized library (by family and die-size) describing a specific power model for each type of module. For example, the power model of a flip-flop is given by

$$P = P_{CK} \cdot F_{CK} + P_{DOUT} \cdot F_{DOUT} + P_{Din} \cdot F_{Din}$$

where F_{CK} is the average clock-input frequency for this flip-flop, F_{DOUT} is its average data-output frequency, and P_{CK} , P_{DOUT} , and P_{Din} are three constants estimated by electrical simulation and silicon characterization for this flip-flop module.

- For an I/O, the formula used for computing the power consumption depends on the I/O technology and the family. For example, for a TTL output, the dynamic power is given by

$$P = P_{INT} \cdot F + C \cdot V^2 \cdot F$$

where C is the output load of the port (35 pf for TTL), V is the output's voltage swing (3.3 V for TTL), P_{INT} represents an internal power contribution dissipated in the pad, and F is the average switching frequency of the IO.

- For a complex block, like a RAM, a FIFO, or a PLL, SmartPower uses a high-level power model that integrates design parameters.

SmartPower automatically computes all the constant parameters of these equations. However, the frequencies depend on the target frequencies of your design. Since it is impractical to enter each frequency manually, SmartPower has several flows that help you to estimate the frequencies and calculate the power consumption.

Timing Analysis

Welcome to Timer

Timer is Actel's static timing analysis tool. Timing analysis is a convenient and thorough method of analyzing, debugging and validating the timing performance of a design. This is achieved by breaking down the design into sets of paths. Delays for each path are then calculated and every path is checked for timing violations.

You can only use Timer after you open a compiled design (*.adb file), or after compiling a netlist in designer. If you invoke Timer before compiling your netlist, Designer guides you through the compile.

There are three ways to start Timer:

1. Choose Timer from the Tools menu, or
2. Click the Timer icon in Designer's toolbar, or
3. Click the Timer button in Designer's design flow.

Timer Interface

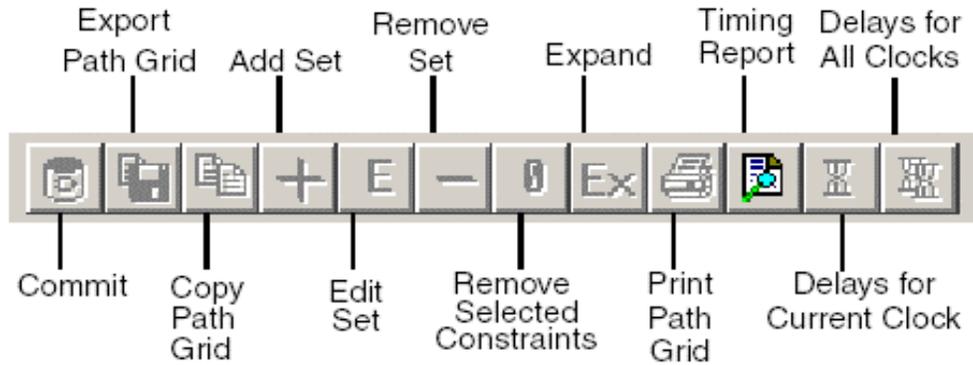
Timer user interface

Timer's four tab screens organize and display static timing information according to the timing analysis preferences you set in the **Preferences** dialog box.

Timer consists of four tab screens: **Summary**, **Clocks**, **Paths**, and **Breaks** (Timer does not display the Clocks tab screen if the device you are using has no clock).

Timer Toolbar

The Timer toolbar contains commands for performing common Timer operations on your designs. Tool tips are available for each button.



Timer Toolbar

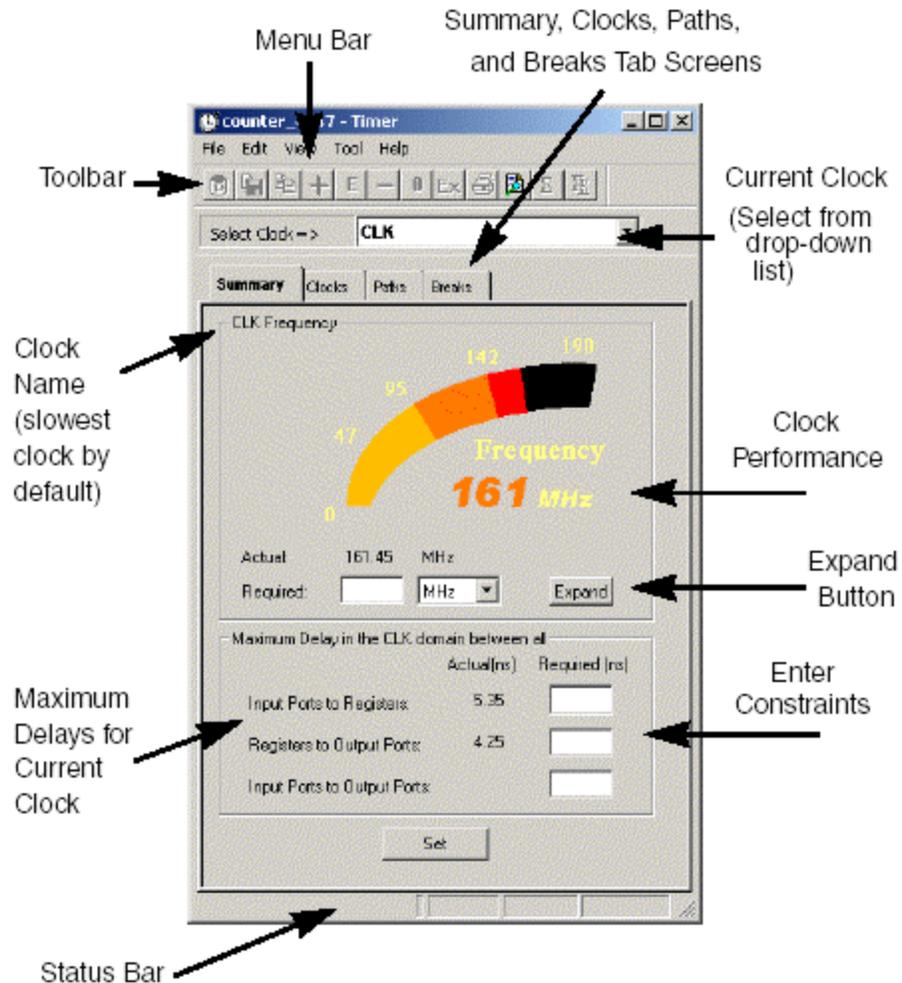
Status Bar

Timer's status bar displays information on menu commands, error messages, your selected temperature, voltage, and speed grade. In addition, Timer displays the following:

- **Temp:** Displays the temperature consistent with the operating conditions selected.
- **Volt:** Displays the voltage consistent with the operating conditions selected.
- **Speed Grade:** The speed grade of the selected device.

Summary tab

By default, Timer's Summary tab screen displays the maximum frequency for the current clock selected in the Select Clock drop-down list box. If you have multiple clock groups or gated clocks, please add timing constraints in Timer to get the correct frequency of your design.



Timer Summary Tab Screen - PC Only

To change the default clock, select one from the Select Clock list.

Click the **Expand** button Timer to display the details of the path that determined the maximum clock frequency in the Expanded Path window.

The Summary tab displays the actual longest/shortest delay between all Input ports to registers, Registers to Output Ports, and Input Ports to Output Ports.

Enter your new delay values in the Required input boxes and click Set to recalculate your delays.

IMPORTANT:By adding constraints to the set of paths listed in this tab, be careful that you do not over-constrain the design. This may degrade the quality of the Timing Driven Layout and increase the overall run time.

Timer Expanded Path Window

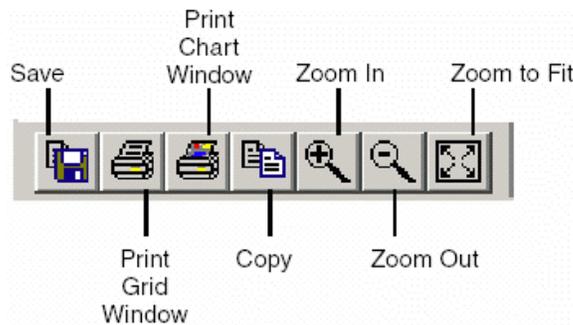
The Expanded Paths window is comprised of three components:

Expanded Paths Grid - Shows all delay components for the selected path (Instance, Net, Macro, Delay, Type, Total Delay and Fanout details). For Delay, (r) stands for rising edge and (f) for falling edge. If you expand a register to register path, the Expanded Path window displays relevant register setup timing information. Click a component to select the corresponding element in the schematic.

Setup Check / Hold Check window - Set the Show option (in Preferences) to "Longest" to view a detailed analysis of the Setup Check. Set the Show option to "Shortest" to view a detailed analysis of the Hold Check. (These analyses include clock insertion delay information.)

Expanded Paths Schematic - Displays a schematic view of the expanded path.

The Expanded Path window includes a toolbar that enables you to Save, Print, Copy, and Zoom in the schematic.



Timer Expanded Path Toolbar

Save: Save the contents of the Grid as a .txt file

Print Grid Window: Print the contents of the Grid window

Print Chart Window: Print the contents of the Chart window

Copy (Grid to Clipboard): Copy the contents of the Grid window to the clipboard

Zoom In: Click to zoom on the Chart window

Zoom Out: Click to zoom out on the Chart window

Zoom to Fit: Click the Zoom to Fit button to automatically fit the entire path in the Chart window

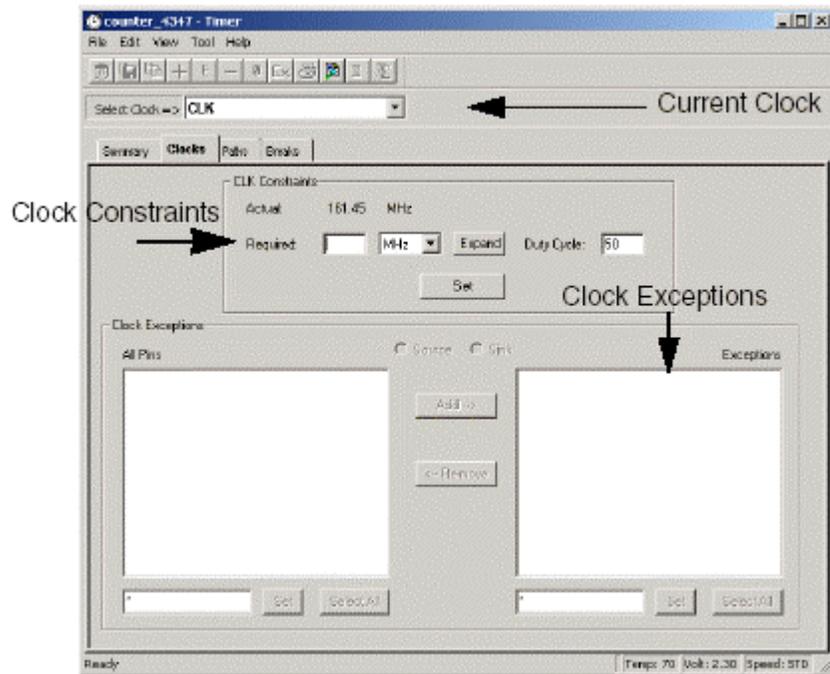
Note:

The expanded paths window for eX, SX-A, Axcelerator, and Flash devices shows the expanded path on a pin-to-pin basis rather than an input-to-input basis. There is a separation between the module delay and the net delay.

Anything you select in the Expanded Paths grid or Schematic window is reflected in both windows.

Clocks tab

The Clocks tab allows you to enter constraint information and set clock exceptions. Select the default clock from the Select Clocks list.



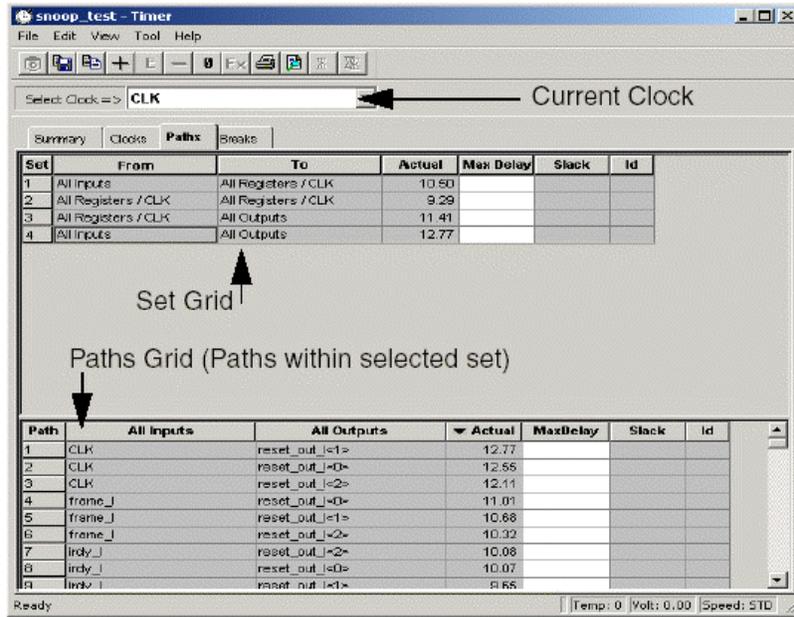
Clocks Tab

Enter constraint information in the **Constraints** area and click **Set**.

Clock exceptions are terminals in a synchronous network that should be excluded from the specified clock analysis.

Paths tab

The Paths tab displays timing analysis information for categories of paths, known as "sets," and the paths within each set. The Paths tab displays the sets in the set spreadsheet (at top) and the paths within each set (at bottom).



Paths Tab

The Paths tab default setting displays four path sets:

From All Inputs TO All Registers / CLK

All paths from the input ports of the design to the input pins of all the registers in the current clock domain. In this instance, CLK is an example of the current clock domain.

From All Registers / CLK TO All Registers / CLK

All paths from the clock, clear, and pre pins of registers in the current clock domain to the input pins of all the registers in the current clock domain; in this instance, CLK is an example of the current clock domain. To view the register setup and clock skew, right-click the desired path in the paths grid and select Expand Path from the shortcut menu, or click the Expand Path button.

From All Registers / CLK TO All Outputs

All paths from the clock pin of registers in the current clock domain to the primary outputs of the design.

All Inputs TO All Outputs

All input ports to all output ports in the design. This set is completely asynchronous (independent of the clock).

All the sets default to display the longest path in the category. You can change this default setting by selecting Preferences from the File menu. When you select a set, Timer displays the paths within the set in the lower spreadsheet labeled "Paths." The spreadsheet displays a sorted list of paths (the number of paths it displays is controlled in the Preferences dialog box). Double-click the column headings to sort the columns.

Note: The run-time required to compute the content of the spreadsheet is a function of the number of paths you wish to display. Select Preferences from the File menu to change the default settings.

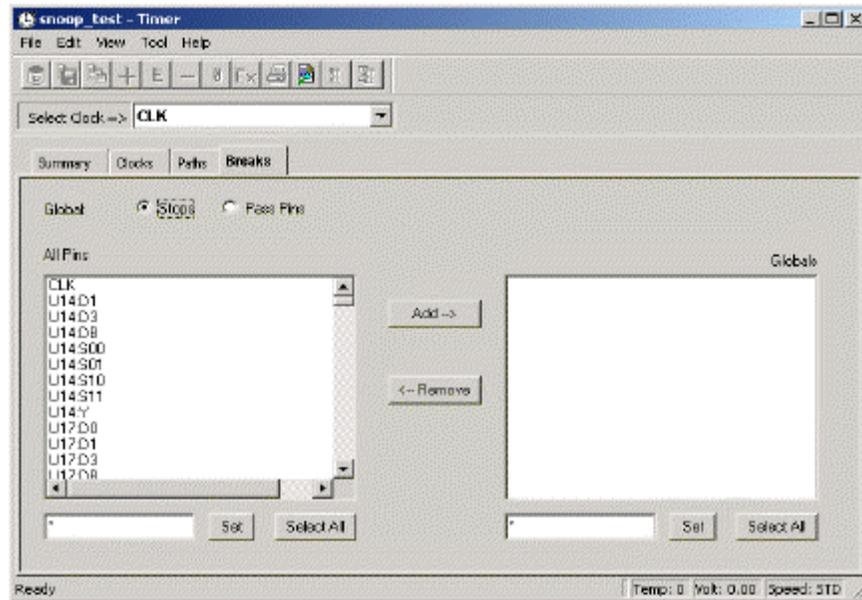
The timing information displayed for sets and paths includes:

- **Actual:** The actual delay calculated by Timer for each path.
- **Slack:** The difference between the maximum required delay and the actual delay.
- **Max Delay:** The maximum required delay specified. Do not interpret this value as the clock frequency. To set clock frequency, input on the Summary tab, or on the Clocks tab.
- **ID:** The constraint ID for the path.

Breaks tab

Use the Breaks tab to enter global stops and pass pins. A global stop is a defined intermediate point in a network that forces all paths through the defined point to be "don't care" paths regardless of any constraint assignment. Setting a pass pin on a module pin enables you to see a path through individual pins, which you are not normally allowed to view a path through.

Note: If you are not careful when you set a pass pin on a module pin, you may set a false path in your design.



Breaks Tab

Timer Menu Commands

The PC and workstation versions of Timer have the same menus. However, some dialog boxes may look slightly different on the two platforms due to the different window environments. The functionality is the same on both platforms, though the locations of the fields and buttons on the dialog boxes may vary. The names of some fields may also vary between the PC and workstation versions.

File Menu

Commit: Commits timing information to Designer

Export Set Grid: Exports selected cells in the sets grid to a file

Export Path Grid: Exports selected cells in the path grid to a file

Print Set Grid: Prints selected cells in the sets grid

Print Path Grid: Prints selected cells in the paths grid

Operating Conditions: Displays the operating conditions Timer uses to calculate delays. Operating conditions change depending on your Worst, Typical, or Best case selection in the **Preferences** dialog box.

Preferences: Invokes **Preferences** dialog box, where you can set analysis and display preferences

Close: Closes Timer

Edit Menu

Copy Set Grid: Copies set grid to clipboard

Copy Path Grid: Copies path grid to clipboard

Remove Selected Constraints: Removes selected constraints, not all

Remove All Constraints: Removes all constraints in Timer

Expand Paths: Expands path in new window

Add Set of Paths: Defines and adds new path set to Paths tab

Edit Set of Paths: Edit added path set

Remove Set of Paths: Removes added path sets

Tool Menu

Report Paths: Generates Timing report

Report Violations: Generates a Timing report, timing violations only

Calculate delays: Calculates delays for the current clock

Calculate all delays: Calculates delays for all clocks and selects the worst (clock with greatest delay)

Help Menu

Help Topics: Lists of Help Topics

Reference Manuals: Opens Timer's User's Guide

Timing report dialog box

External Setup-hold Timing Check: Selecting the External Setup hold timing checkbox adds specific sections to the timing report, including External setup and hold as well as clock-to-out timing information.

Slack Threshold (ns): Sets the maximum slack threshold for all the paths included in the report. Use this value in conjunction with the **Sort by: Slack** option in the Timer **Preferences** dialog box.

Sort by slack (in Preferences): If you select Sort By Slack in the Preferences dialog box, you can also limit the number of delays displayed based upon the Slack threshold.

For example, if you want to see only the delays which have a slack less than 5 ns, select Slack in the Sort by drop-down list box in the Preferences dialog box and then enter 5 in the Slack Threshold text box in the Timing Report dialog box. The timing report displays all the timing paths that have a slack of 5 ns or less (that is, all paths that met the timing constraints by ≤ 5 ns as well as all the paths that failed to meet constraints).

If you wish to display the timing paths that failed by 10 ns or more, enter -10 in the Slack Threshold box; the timing report displays the paths that failed to meet the timing constraints by 10 ns or more.

Calculating Delays

Delays, PLLs, RAMs, and FIFOs

Actel Timer uses two different timing models, "pin-to-pin" and "input-to-input". The first type uses a "pin-to-pin" timing model, because Timer reports a pin-to-pin delay. The second type uses an "input-to-input" timing model, because Timer reports the delays from an input gate to the input of the next gate by lumping the gate and net delays together.

ACT1, ACT2, ACT3, DX, MX, and SX devices use the input-to-input timing model, while the 54SX-A, RTSX-S, eX, Axcelerator, and Flash devices use the pin-to-pin timing model. Some timing analysis features are specific to the different timing models; exceptions are noted in the help.

The delay for pin-to-pin devices is reported until the input pins of the registers. Therefore, setup time is not included in the delay. However, the register setup and hold, as well as the clock skew, are taken into account during the analysis of setup check and hold check when identifying timing violations. Setup, hold, and clock skew are also taken into account during clock frequency estimation.

For information on the setup and hold process in Timer, see the Expanded Paths window. It enables you to view clock network insertion delay and clock skew information.

PLLs

The timing tool sees a PLL as a register and a clock generator. Any clock output port in a PLL is a potential clock (and appears in the list of potential clocks for the design). Like all other potential clocks, you can constrain these PLL output clocks by setting any clock constraint independently. The input clock of the PLL on which you set the constraint is not the clock input port of the PLL but the clock driving this clock input port. The driving clock will be a Primary port of the design, a register's output, or another PLL's output.

PLLs for Axcelerator

By default, when you set a clock constraint on the clock source connected to the clock input of the PLL, Timer automatically computes the clock constraints on the outputs of the PLL (according to the PLLs configuration). Thus, the value of the clock output is equivalent to the clock input multiplied, divided, or shifted by the value of your static configuration.

If you specify a clock constraint for the output clock(s), the PLL ignores the static configuration value and delivers a clock frequency according to your constraints. Timer reports this value accurately. In addition, if you remove your constraints on the output clock(s), the Timer tool recalculates your output frequency according to your static configuration value. For more information on generating PLLs and their logic characteristics, please refer to the *ACTgen Macros Reference Guide* in the online help or in .pdf format.

Note: For ProASIC^{PLUS} devices, the PLL is only considered as a register; there is no output clock computation.

RAMs and FIFOs

The Timer tool displays blocks of RAM and FIFO as a single “black box,” (you have as many black boxes as you have instantiations of RAMs and FIFOs in your design). Thus, if you construct a RAM or FIFO cell out of several RAM blocks, Timer sees and treats the cell as a single black box. Timer does not display timing information within individual black boxes, because all the delays are reported using the interface of the RAM. Timer displays timing information between black boxes and other logic in the design.

Timer treats RAMs and FIFOs as registers, and like any register, they have clock signals. For more information about RAMs and FIFOs, please refer to the *ACTgen Macros Reference Guide*, in the online help or in .pdf format.

FIFO: Timer displays the paths to the FIFO flags depending on their clock. Timer shows paths to Empty and Almost Empty with respect to the Read clock; paths to Full and Almost Full are displayed with respect to the Write clock.

Using Timer

Determining your clock frequency

Because a design's performance is often measured through the clock frequency, determining the clock frequency is the most common use of static timing analysis.

To obtain a specific clock frequency:

1. Click the **Summary** tab in the **Timer** window.
2. Select a clock from the **Select Clock** list. The selected clock becomes your current clock. The frequency is displayed under the speedometer.

The clocks listed in the pull-down menu are defined as signals which drive the clock or gated input of two or more adjacent registers. For pin-to-pin delay families, one register is enough to have the clock listed as a potential clock. Timer does not support virtual clocks.

In frequency calculations, values for latency is assumed to be 0, the duty cycle is 50%, and the clock edge is rising. (You can set the duty cycle in the **Clocks** tab.) For pin-to-pin timing model families, Timer takes into account the register setup and the clock skew when estimating the maximum clock frequency. However, the setup value is not included in the actual delay reported between the clock pin of a source register and the data pin of a sink register. For more information on calculating delays, please refer to [Calculating Delays](#).

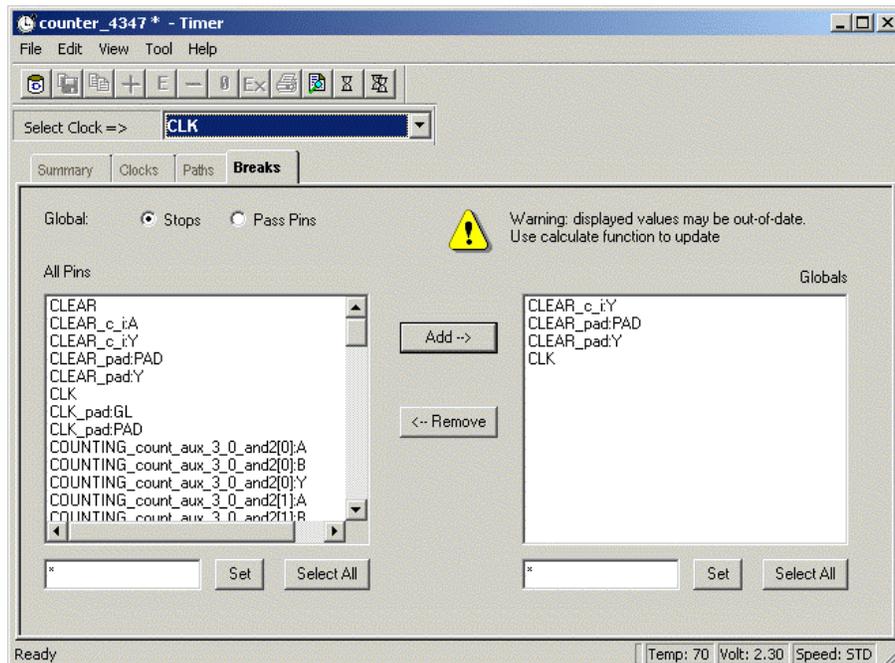
Adding and Removing Break Points

The Timer default behavior is to break paths at clocks. You can change this default behavior in the **Timer Preferences** dialog box. Without stop points (or break points), Timer reports all the paths from pad to pad in the design. If you do not want to see the paths going through registers clock pins, you could specify these as break points. The path running through those pins is not displayed.

Setting a pass on a module pin enables you to see a path through individual pins. Additionally, you can set a global pass on all Clk/G and Clr/Pre pins in the Preferences dialog box, which is available by choosing Preferences from the File menu.

To add break points:

1. Click the **Breaks** tab.
2. Select **Global Stops** or **Pass Pins**. The **All Pins** list box displays the pins.
3. Select the pin(s). The **All Pins** list box defaults to show all pins. Text boxes are provided below the list boxes to help you limit the list for consideration. Enter a value and click **Set**. The * character is used as a wildcard. To select multiple pins, hold down the CTRL key while selecting with your mouse. Click **Select All** to select all pins displayed in the **All Pins** list box.
4. Click **Add**. The **Stops** or **Pass Pins** will be added to the **Global** list box as break points.



Breaks Tab - Stops Selected

To remove break points:

1. Click the **Breaks** tab.

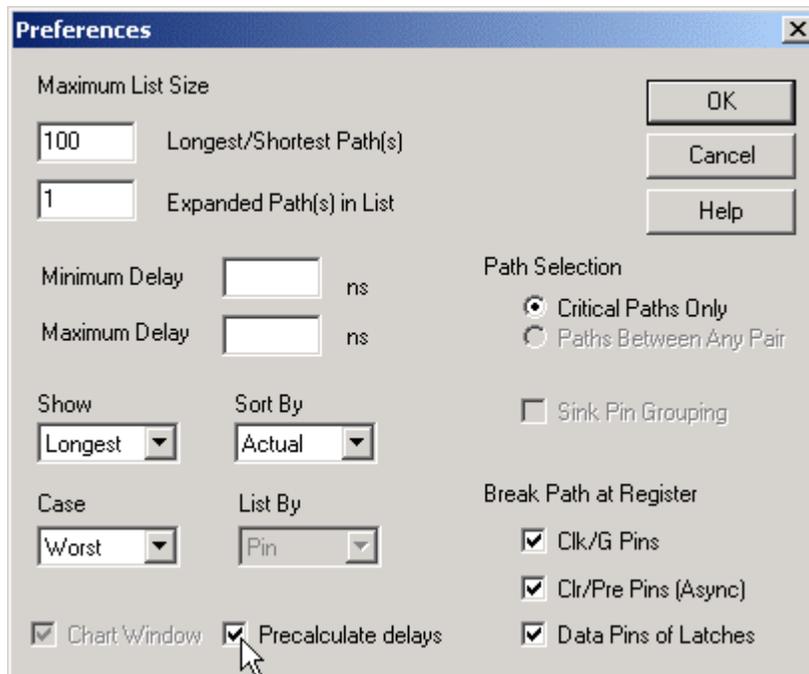
2. Select **Global Stops** or **Pass Pins**. The break points are displayed in the Global list box.
3. Select break **Points** to remove. To select multiple breaks, hold down the CTRL key while selecting with your mouse. Click **Select All** to select everything displayed in the **Globals** list box.
4. Click **Remove**. The pin(s) will be removed from the **Globals** list box.

Setting Preferences in Timer

Delay preferences

The Preferences dialog box controls your delay and timing analysis preferences. If you are using UNIX, consult the UNIX Command Summary for a list of the commands related to preferences.

You may wish to prevent Timer from calculating delays when the tool starts. (By default, Timer estimates all potential clock frequencies, as well as the 4 delays corresponding to the 4 primary sets (All inputs to All outputs, All inputs to registers, registers to All outputs, registers to registers) associated with the slowest clock when the tool opens.)



Preferences - Precalculate Delays Selected

To change your option for pre-calculating delays, from the **File** menu, select **Preferences** and deselect "Precalculate delays." Alternatively, you may choose to modify the related variables. To do so, from the Designer GUI, in the Options menu, select "Set Variable." In the Variable Name dialog box, enter:

TIMER_PRECALCULATE_DELAYS

And in the Value dialog box, enter

0

(default is "1").

Changing and displaying paths

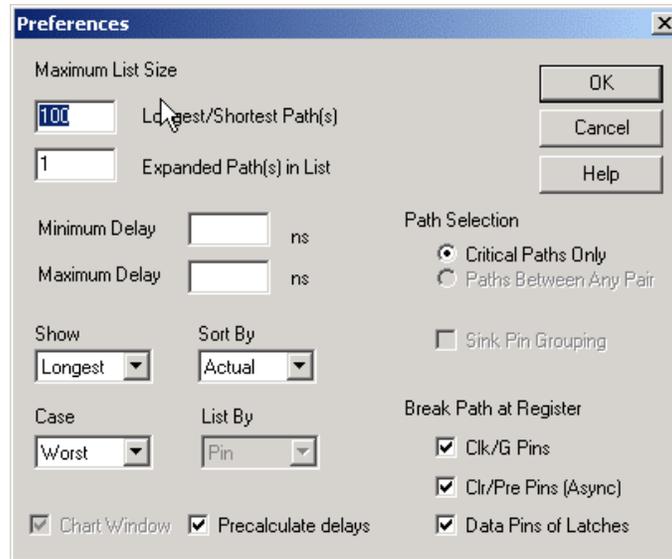
Use the **Preferences** dialog box to control the number of paths displayed in the **Paths** tab, **Expanded Paths** window, and **Timing Report**.

To change the number of paths that Timer displays:

1. In the **File** menu, click **Preferences**. The **Preferences** dialog box appears.
2. In the **Maximum List Size** area, enter your default preferences for the maximum number of **Longest/Shortest Path(s)** and **Expanded Path(s)** that you want to display

The value for **Longest/Shortest Path(s)** is set to 100 by default; you can modify the value if you wish to see more paths in your report. However, the higher the value the longer it takes to invoke Timer.

3. Click **OK**.

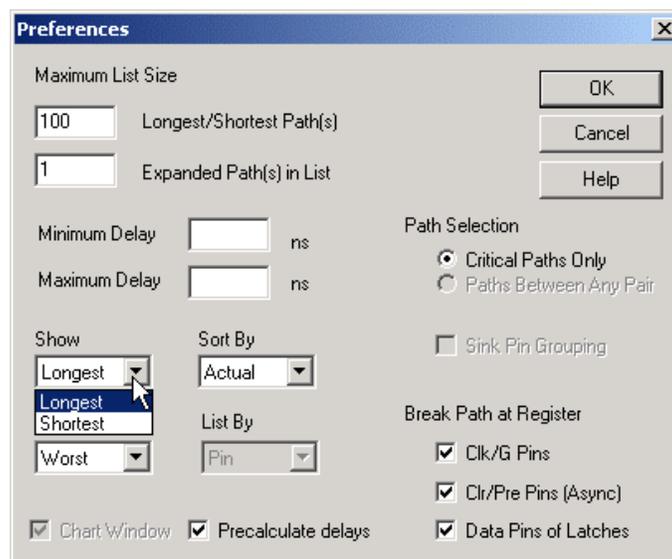


Displaying the Shortest Paths First

By default, Timer displays the first 100 paths from longest to shortest in the **Paths** tab and **Expanded Paths** window. When Timer displays paths from longest to shortest, it reports setup times for registers. To view hold times in the **Expanded Path** window, you must set the **Preferences** to **Show** the **Shortest** paths.

To display the shortest paths first:

1. In the **File** menu, click **Preferences**. The **Preferences** dialog box appears.
2. Select **Shortest** from the **Show** drop-down menu.
3. Click **OK**.



Setting Preferences in Timer to Display Shortest Paths

Delay filters (max. or min.) / Sorting by actual or slack delays

Setting Minimum or Maximum Delay Filters

Use the **Preferences** dialog box to filter paths for delays above, below, or between a specified value. Enter your display preferences in the **Maximum Delay** and **Minimum Delay** boxes and click **OK**.

Sorting and Displaying by Actual or Slack Delays

Timer can display delay information in two ways:

- Actual delay values
- Slack, which is the difference between actual delay and a user-specified delay (that is, user-specified constraint)

Displaying by Actual Delay

The actual delay is the path delay between two points in your design. This is the only way to sort your data if you do not have any timing constraints entered (for information on setting timing constraints, see *Constraint Guidelines*). If you have entered timing constraints, the actual delay report automatically displays the slack - even if you don't ask for it - but the data will always be listed from longest to shortest actual delay.

Actual delay measurements may be calculated before or after layout.

To display Actual delay:

1. In the **File** menu, click **Preferences**. This displays the **Preferences** dialog box
2. Select **Actual** in the **Sort By** pull-down menu.
3. Click **OK**.

Displaying by Slack Delay

Slack delay is the delay difference between a timing constraint entered in Timer and the actual delay of a path. For example, if a signal takes 20 ns to get from point A to point B, and you entered a timing constraint of 15 ns, the Timing Report would list -5 ns slack for that path. Thus, if the slack is negative, then the actual delay did not meet the desired timing by the absolute value of the slack (in ns). Conversely, if the slack value is positive, then the timing constraint was met, with the slack value (in ns) to spare. In a violations report, Timer sorts the data (by default) from longest to shortest slack.

When displaying slack, all the paths without timing constraints are filtered from the reported data. This enables you to quickly determine how well your design meets your timing requirements. This is especially useful for viewing critical delays like register-to-register, clock-to-out, and input-to-register.

Best\Typical\Worst Case Analysis

By default, Timer displays the worst case analysis.

To display the best or typical case analysis:

1. In the **File** menu, click **Preferences**. The **Preferences** dialog box appears.
2. Select **Best**, **Typical**, or **Worst** from the Case drop-down menu. If you change the setting, timing is recalculated for the entire design; this may take a few minutes.
3. Click **OK**.

Selecting paths - Adding or removing break paths

Normally, Timer displays only critical paths. Critical paths are the longest path between any of the starting points (terminals) and each ending terminal. If you would like to see the timing of all paths between any of the starting terminals and any of the ending terminals, select **Paths Between Any Pair** (input-to-input timing model families only) in the **Path Selection** area of the **Preferences** dialog box. Selecting **Critical Paths** displays only critical paths.

Adding and Removing Break Paths

Asynchronous feedback paths in a design can cause paths to be reported as having excessive delays. The most common example is feedback paths through asynchronous Set or Reset pins to banks of flip-flops, like a state machine or a counter.

To exclude paths:

1. In the **File** menu, click **Preferences**. The **Preferences** dialog box appears
2. **Break Paths at Register**. Choose **Clk/G Pins**, **Clr/Pre Pins (Async)** or **Data Pins of Latches** to prevent displaying paths that pass through either clock, gated, clear, preset, or data pins of flip-flops or latches.

Note: The **Break Paths at Register** option is selected by default, and the paths are excluded. Deselect the checkboxes in the **Timer Preferences** menu to display these paths.

3. Click **OK**.

Timer UNIX Preferences

For UNIX users, the summary of commands for Timer is as follows:

```
report -type timer
[-sortby {actual,slack}]
[-maxpaths <num>]
[-case {worst,typical,best}]
[-path_selection {critical,any_pair}]
[-setup_hold {on,off}]
[-expand_failed {on,off}]
[-clkpinbreak {on,off}]
[-clrpinbreak {on,off}]
[-latchdatapinbreak {on,off}]
[-slack <num>]
```

Path Analysis

Timer organizes and displays data based on your timing analysis preferences. Timer assists you in analyzing critical paths, paths with the greatest delay, and by expanding paths so you can trace delays along paths. The Expanded Path window provides delay information for the path that is in greatest violation.

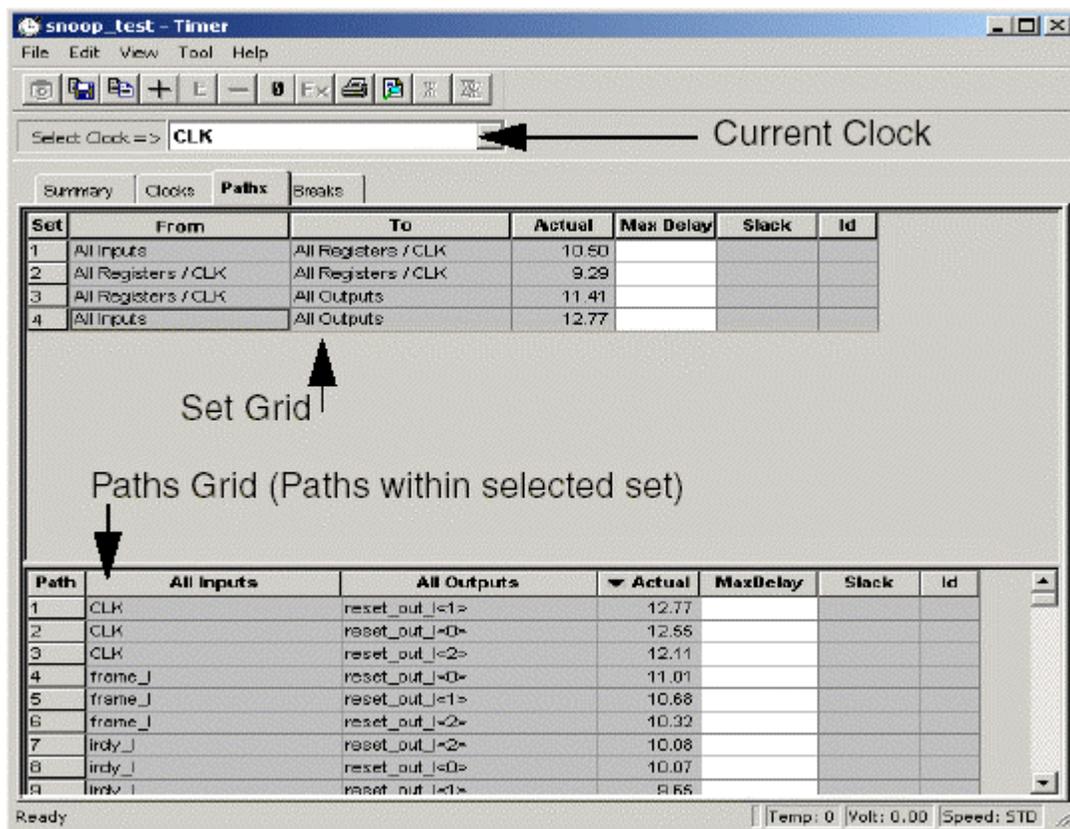
You can change your preferences to control how you display paths, add path sets, and expand paths.

Display paths

Path sets (groups) and paths within each set are displayed on the Paths tab. You can create your own sets and add them to the paths tab (see Adding path sets). Also, Timer displays all previously entered sets that have a constraint in the Set grid.

To display paths:

1. Click the **Paths** tab. By default, Timer displays four path sets in the set grid.
2. Click a set. Timer displays the paths in the path grid.



Timer Paths Tab

When you set a clock constraint for a pin-to-pin timing model family, it is mapped into specific register to register max delay values; these values appear in the max delay of each specific path in the spreadsheet. Timer takes into account register setup and clock skew when computing max delay values for these pin-to-pin model families.

The register-to-register selections are based on the clock domain selected in the Clocks tab. (To select another clock, choose the Select Clock menu from the Toolbar.) See the index for a list of Paths tab information.

All delays shown are worst-case by default. To change this setting, see *Case Analysis* in the index.

Expanding paths

Each path has one or more logic macros and nets that contribute to its total delay. By expanding the path, you can view detailed delay information for parallel paths. Note that with the exception of parallel edges, parallel paths are not available for families that use the pin-to-pin timing model.

To expand a path:

1. Click the **Paths** tab.
2. Select a path set. Paths within that set are displayed in the path grid.
3. Select the path you wish to expand.
4. Expand the path by double-clicking the path, right-click and select **Expand Path** from the shortcut menu, or in the **Edit** menu, click **Expand Path**. The **Expanded Path** window opens and displays a single path in the **Expanded Path Grid** and a graphical representation of the paths in the **Schematic Window**.

Grid2	Instance	Net	Macro	Delay	Type	Total	Fanout
1	count_aux[3]:D	COUNTING_co	ADLIB:DFF	0.32 (f)	Net	5.43	0
	COUNTING_count_aux_		ADLIB:AND	0.70 (f)	Cell	5.11	1
	COUNTING_count_aux_	G_34	ADLIB:AND	0.25 (f)	Net	4.41	0
	G_34:Y		ADLIB:XOR	0.63 (f)	Cell	4.16	1
	G_34:B	G_32	ADLIB:XOR	0.19 (r)	Net	3.53	0
	G_32:Y		ADLIB:NAN	0.44 (r)	Cell	3.34	1
	G_32:A	G_29	ADLIB:NAN	0.27 (r)	Net	2.89	0
	G_29:Y		ADLIB:NAN	0.45 (r)	Cell	2.62	2
	G_29:A	G_28	ADLIB:NAN	0.27 (r)	Net	2.17	0
	G_28:Y		ADLIB:NAN	0.75 (r)	Cell	1.90	2
	G_28:B	count_aux_c[0]	ADLIB:NAN	0.33 (f)	Net	1.15	0
	count_aux[0]:Q		ADLIB:DFF	0.82 (f)	Cell	0.82	3
	count_aux[0]:CLK	CLK_c	ADLIB:DFF	0 (r)	Net	0.00	0

```

data path delay                5.43      7.33
data arrival time              7.33

clock CLK                      6.15      6.15
clock network delay(shortest)  1.90      8.05
library setup time            -0.77      7.29
data required time            7.29

-----
data required time            7.29
data arrival time            7.33
slack (VIOLATED)             -0.04
    
```

Timer Expanded Path Window

Clock frequency should be the inverse of reg-reg delay plus setup time. However, this is not true with clock skew.

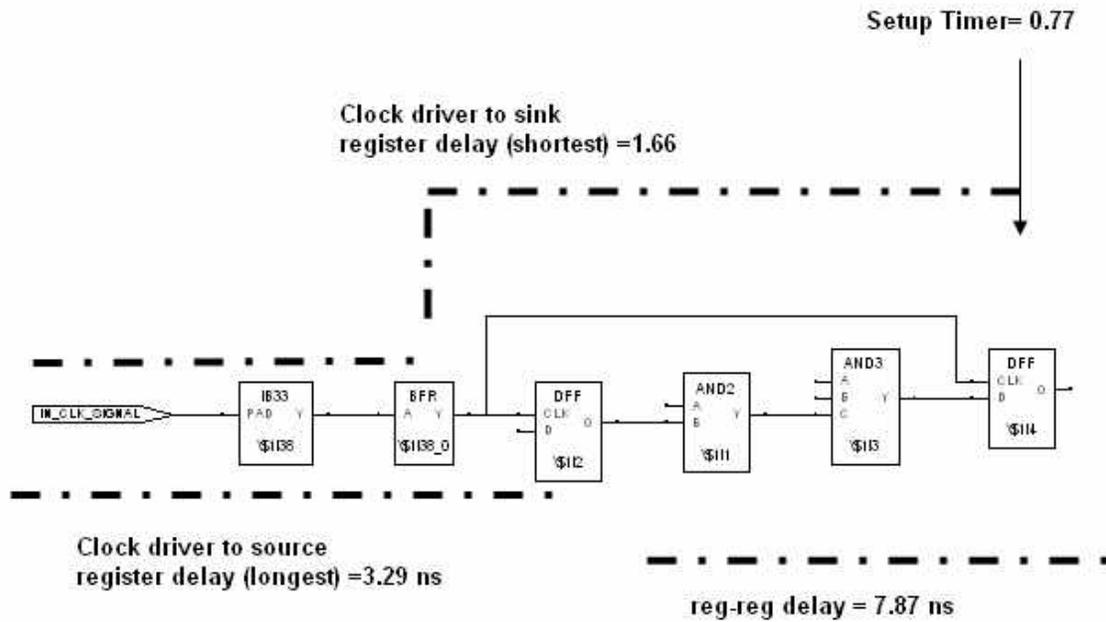
For pin-to-pin timing model families (54SX-A, eX, Axcelerator, and Flash devices), Timer takes into account the register setup and the clock skew (starting in R1-2003) when estimating the maximum clock frequency. However, the setup value is not included in the actual delay reported between the clock pin of a source register and the data pin of a sink register. The Timer Expanded Path window shows Setup Check / Hold Check information. Set the Show option (in Preferences) to "Longest" to view a detailed analysis of the Setup Check and set the Show option to "Shortest" to view a detailed analysis of the Hold Check. (These analyses include clock insertion delay information.)

The images below show the correlation between the Expanded Path Grid (at top), the Setup Check or Hold Check report (at middle) and the Schematic window (at bottom).

Note that the values displayed in the Setup Check / Hold Check report are affected by rounding in calculations, so you may see a very slight discrepancy in the difference between the data required time and the data arrival time (the slack time).

Grid1	Instance	Net	Macro	Delay	Type	Total	Fanout
1	\$114:D	\$1N9	ADLIB:DFF	1.18 (f)	Net	7.87	0
	\$113:Y		ADLIB:AND	0.44 (f)	Cell	6.68	1
	\$113:C	\$1N7	ADLIB:AND	2.51 (f)	Net	6.25	0
	\$111:Y		ADLIB:AND	0.73 (f)	Cell	3.74	1
	\$111:B	\$1N5	ADLIB:AND	2.20 (f)	Net	3.01	0
	\$112:Q		ADLIB:DFF	0.82 (f)	Cell	0.81	1
	\$112:CLK	clk1	ADLIB:DFF	0 (r)	Net	0.00	0

Name	Setup Check	Delay (ns)	Total (ns)
clock IN_CLK_SIGNAL		0.00	0.00
clock network delay(longest)		3.29	3.29
data launch time			3.29
data path delay		7.87	11.16
data arrival time			11.16
clock IN_CLK_SIGNAL		8.33	8.33
clock network delay(shortest)		1.66	9.99
library setup time		-0.77	9.22
data required time			9.22
data required time			9.22
data arrival time			11.16
slack (VIOLATED)			-1.93



$$\text{Slack} = \text{data required time} - \text{data arrival time} = (8.33+1.66-0.77)-(3.29+7.87) = -1.93\text{ns}$$

Clock Skew Analysis

The difference in the arrival times of the clock signals between two sequentially-adjacent registers (clock skew) may cause a design to malfunction with short data paths. The most efficient method to eliminate the short data path problem is to minimize the clock skew by using the low-skew global routing resources for clock signals.

Please refer to the Actel website (<http://www.actel.com>) for an application note on clock skew analysis.

To measure clock skew:

1. **Specify clock frequency.** In order to obtain hold margin calculations, Timer requires that you specify a clock frequency. Timer uses the frequency to calculate the period, which it needs to evaluate the margin for adjacent flip-flops with alternate clock edges. If you do not enter a clock frequency in the summary tab, you will not get any results.

To enter a frequency, select the desired clock under “select clock”, and enter a frequency under “required” in the Summary tab. Click **Set** when you are done.

2. **Set operating conditions.** To measure clock skew, perform hold time analysis for **BEST** case in **Shortest** path mode. Set the **Case** in **File -> Preferences**.
3. **Run Violations Report.** A report is available from Timer that provides a summary of timing margins for all paths in the design. In Timer, go to **Tool -> Report Violations**. This report lists the following categories:
 - Section Clock constraints violation
 - Section Max Delay constraints violation
 - Section Min Delay constraints violation

To find a summary of hold time margin in your design for the given operating conditions, refer to the timing paths listed under **Section Min Delay constraints violation**.

The first column defines the slack for each path. Positive values represent margin, negative values represent a violation.

4. **Detailed analysis.** To see the details of a given path, go to the **Paths** tab in Timer. You can look for a specific path by creating a new path set for the specific path(s) you are interested in. To create a path set, go to **Edit -> Add set of paths**. Refer to the **Paths** tab for information on how to add a set of paths.

Once you have defined the new path set, click the set to display the path list in the lower spreadsheet. Then highlight the path you are interested in, right click and select **Expand Path**.

5. **Review Expanded Path window.** The expand path window shows details of the calculations performed in the clock skew analysis.

The margin is calculated by adding the clock propagation delay of the master register to the data path delay between the two registers. This is the data arrival time. Then the clock propagation to the slave register is subtracted from the sum, giving the final slack value. If alternate clock edges are used for adjacent registers, Timer considers the clock period accordingly.

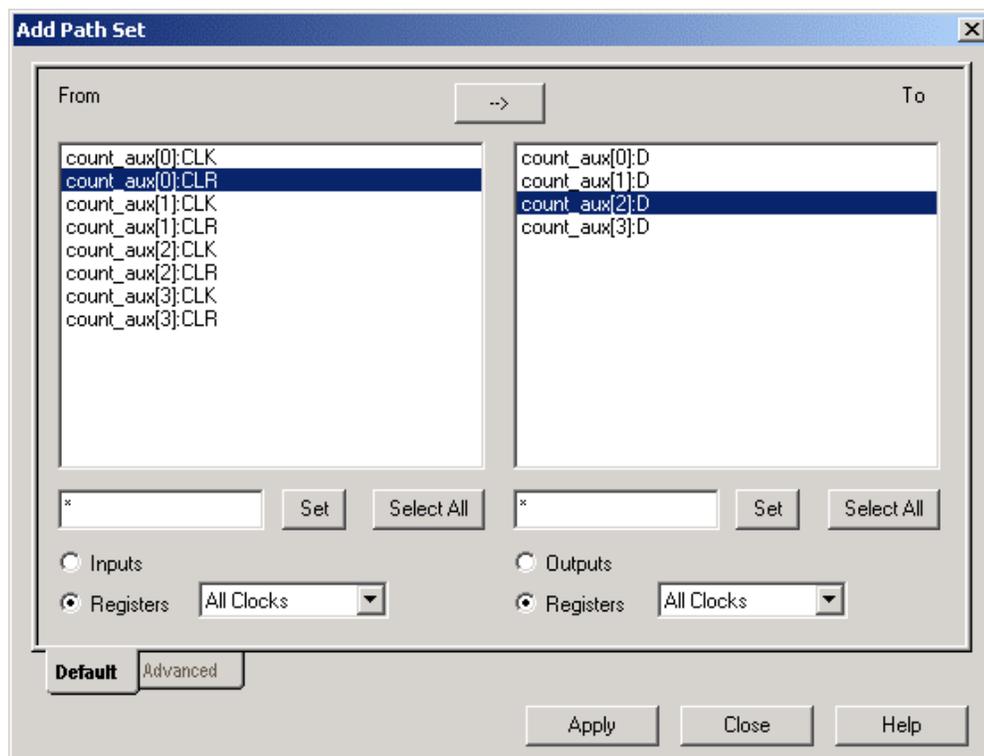
Adding path sets

Create and add path sets to the Paths tab to determine delay information and enter constraints. User-defined sets enable you to customize the sets that are available for analysis. By creating custom sets, you can simplify timing analysis and constraint setting for specific blocks or paths in your design.

For example, if you are concerned about timing of the lower-level block “sub_block_1” in your design, you can create a set that only includes timing paths in that block.

To add a set:

1. Click the **Paths** tab.
2. From the **Edit** menu, select **Add Set of Paths**. The **Add Path Set** dialog box consists of two screens, **Default** and **Advanced**. The **Advanced** tab enables you to use keywords to create a set. For more information on using keywords consult the **Index**.



Add Path Set Dialog Box

3. Select the desired clock.
4. Click the directional button to select path direction.
5. Select the desired **Inputs** (all input pad pins) or **Registers** (all input pins on the flip-flops and latches).

Use the pull-down menus to choose the active clock nets. Choose **All Clocks** for both to find delays for all register-to-register paths. Choose the **Outputs** radio buttons to filter the **From** and **To** list boxes (it limits the **From** and **To** list boxes to all output pad pins).

Select your desired starting and ending points in the **From** and **To** list boxes. Naming filters are provided to limit the list of terminals in the display. The naming filters use the * character as a wildcard and the / character to delimit levels of hierarchy.

For example, use * to filter for all terminals; *:E to filter for all terminals with pin E; U1/* to filter for all terminals in block U1; and U1/*:E to filter for all terminals in block U1 with a pin E. You can also use multiple wildcards such as */U1/*:E. After entering your naming filter, click the Set or Select All button.

If the directional button is pointing right:

1. Select a starting point in the **From** list. The **To** list box displays all corresponding endpoints.
2. Select one or more endpoints in the **To** list box that complete the path set. Click the **Select All** button to select all endpoints.

If the directional button is pointing left:

1. Select the endpoint in the **To** list box. The **From** list box displays all corresponding starting points.
2. Select one or more starting points in the **From** list box. Click the **Select All** button to select all.

Click **Apply** to add the path set to the **Paths** tab. Continue creating and adding sets. When you are done, click **Close** to close the **Add Path Set** dialog box.

Add a "one input to all outputs" path set

To show one input to all outputs, you must add the set to the **Paths** tab. You can then view delay details and set constraints.

To show one input to all outputs:

1. In the **Paths** tab, choose **Add Set of Paths** from the **Edit** menu. The Add Path Set dialog box appears.
2. Select the **Inputs** and **Outputs** radio buttons.

Make sure the directional arrow is pointing to the right, from **Inputs** to **Outputs**. Click the arrow to change its direction.

3. Choose the desired input starting point in the **From** list box.
4. Click the **Select All** button to select all outputs.
5. Click **OK**. The set showing one input to all outputs is added to the **Paths** tab.

Add an "all inputs to one output" path set

To show all inputs to one output you must configure and add the set to the **Paths** tab.

To create a set showing all inputs to one output:

1. In the **Paths** tab, from the **Edit** menu, click **Add Set of Paths**. The Add Path Set dialog box appears.
2. Select the **Inputs** and **Outputs** radio buttons.
3. Click the directional arrow to point it left.
4. Choose the desired output endpoint in the **To** list box.
5. Select all inputs (all starting points), by Clicking on the **Select All** button under the **From** list box.
6. Click **OK**. The set showing all inputs to one output is added to the **Paths** tab.

Edit or Remove a Path Set

To edit a path set:

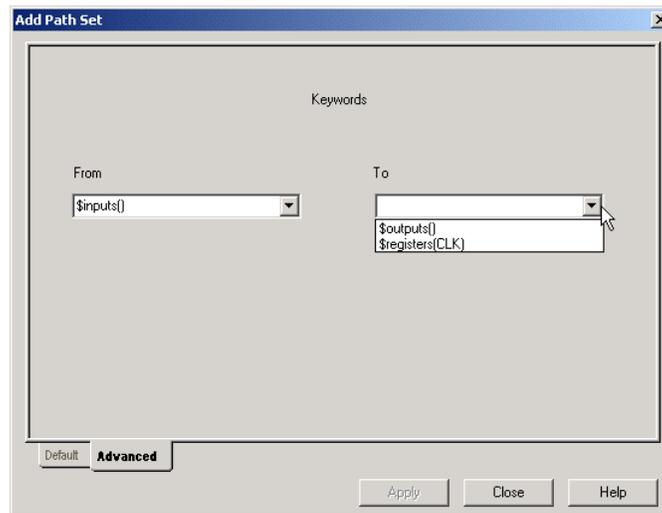
1. Select the set in the **Paths** tab.
2. In the **Edit** menu, click **Edit Set of Paths**, or right-click and choose **Edit Set**.
This displays the Edit Set dialog box.
3. Edit the **Path Set** and click **OK**.

To remove a path set:

1. Select the set in the **Paths** tab.
2. In the **Edit** menu, click **Remove Set of Paths**, or right-click and choose **Remove Set** from the shortcut menu. The set is removed.

Adding/Removing sets with keywords

The **Advanced** tab in the **Add Set** dialog box enables you to use keywords (macros that represent various sets of terminals) to create a set. Timer does not save constraints on the **Advanced** tab in the **Add Set** dialog box. You must **Commit** your changes in the main window to save your constraints.



Add Path Set - Advanced Tab

Keywords for the SX-A, eX, Axcelerator, and Flash families are explained in the Keyword Filters section.

Supported keywords include:

- `$inputs()`

All input and bi-directional pins.

- `$outputs()`

All output and bi-directional pins.

- `$registers(clock_name)`

The pins of all registers driven by the clock whose name is `clock_name`.

To create a set using keywords:

1. Click the **Paths** tab.
2. In the **Edit** menu, click **Add Set of Paths**.
3. Click the **Advanced** tab.
4. Enter the **From** keyword or choose a keyword from the **From** drop-down list box to define the **From** set.
5. Enter the **To** keyword or choose a keyword from the drop-down list box to define to set.
6. Click **OK**. This displays paths for the keyword set in the paths tab.

Timing constraints

Timer enables you to specify timing constraints and requirements for clocks and paths. Use these constraints to generate timing reports and in Timing Driven Layout. In order to run timing driven layout, you must enter and commit your constraints in Timer before you exit the tool.

For Flash, some of the constraints must be entered via GCF and SDC constraints.

For Axcelerator, you can run timing driven place-and-route even if you have not set any user constraints.

Constraint guidelines

Constraint Guidelines

Delay constraints control the Timing Driven Layout engine. You can define these constraints using Timer or by importing an external DCF, SDC, or GCF (for Flash) file. The timing-driven layout engine considers the defined delays when allocating silicon resources with the goal of meeting or beating all constraints if possible. The timing-driven layout engine evaluates the performance criticality of one function versus another when allocating device resources. Because resources are limited, use the following guidelines to ensure the defined constraints meet the needs of the design without impairing device resources.

General Guidelines

- **Set Sufficient Constraints** - All constraints for the design should be defined to ensure correct operation of the Timing Driven Layout engine. Timing Driven Layout considers networks that have not been defined as “don't care” paths, which have a low priority for resource allocation. If these undefined paths are actually critical, they may fail to meet performance demands.
- **Avoid Unnecessary Constraints** - Describe “don't care” paths to free high performance device resources. Not defining a path is one mechanism for doing this. However, it is difficult to avoid defining some “don't care” paths, so Designer provides clock exceptions and global stop sets to enhance this capability (see Clock exceptions).
- **Avoid Over-Constraining** - The Timing Driven Layout engine is designed to achieve or exceed the delay constraint defined (less than or equal). Defining a constraint shorter than is actually required for margin can have a negative impact on the performance of the device because of competition for device resources.

Specifying clock constraints

Use the **Clocks** tab to assign values to each clock network in your design.

To assign clock constraints:

1. Click the **Clocks** tab.
2. Select the clock of interest in the **Select Clock** pull-down menu in the toolbar.

3. Specify the timing requirements. In the **Constraints** area, define the **Required** and **Duty Cycle** areas. Select **MHz** or **ns** from the pull-down menu.
4. Click **Set**.

Clock exceptions

Timer enables you to specify global clock constraints. If you have paths that are not required to meet the global constraint (for example, multi-cycle paths), then list them as exceptions. The Clocks Exceptions area on the Clocks tab provides a mechanism for doing this. A terminal specified as a clock exception will cause all paths beginning or ending at this terminal to be unconstrained by the global timing constraint.

To add or remove terminals from the Clock Exception List:

1. Click the **Clocks** tab.
2. Select the **Clock** name from the drop down menu.
3. Enter a constraint in the **Constraints** area and click **Set**.
4. Select **Source** or **Sink** in the **Clock Exceptions** area. The **Clock Exceptions** area displays the Pins. The terminals of the sequential device are displayed using an `<instance_name>:<pin_name>` format.

For example, a DFM with an instance name of `U1\FF1` will have a single source terminal, `U1\FF1:CLK`, and 3 three sink terminals: `U1\FF1:A`, `U1\FF1:B`, and `U1\FF1:S`.

5. Use the **Filter** field to further sort the list of clock pins. Naming filters are provided to limit the list of terminals for consideration. The naming filters use the `*` character as a wildcard and the `/` character to delimit levels of hierarchy.

For example, use `*` to filter for all terminals; `*:E` to filter for all terminals with pin E; `U1/*` to filter for all terminals in block U1; and `U1/*:E` to filter for all terminals in block U1 with a pin E. After entering your naming filter, click the **Set** or **Select All** button. Multiple `*` and `/` characters may be used.

6. **Add** or **Remove** the clock exception. To add a clock exception, highlight the desired entry from the **Clock Pins** list and click **Add**. To remove an exception, highlight it in the **Exceptions** list and click **Remove**.
7. From the **File** menu, select **Commit** to commit changes.

Path constraints - specifying or removing

You can specify a timing constraint on a specific path or groups (sets) of paths.

To specify a timing constraint for a path set:

1. Click the **Summary** or **Paths** tab.
2. Select a **Path** set.
3. Enter the timing constraint. On the **Summary** tab enter the constraint in the **Required** text box and click **Set**. On the **Paths** tab enter the constraint in the **Max Delay** column.

To specify a timing constraint for a specific path:

1. Click the **Paths** tab.
2. Click the corresponding set in the **Set** grid.
3. Select the path in the path grid.
4. Enter the timing constraint in the **Max Delay** column.

Removing Constraints

You can remove all constraints or just selected constraints. To remove all constraints choose **Remove All Constraints** from the **Edit** menu.

To remove select constraints on the Paths tab:

1. Click the **Paths** tab.
2. Select the path set with the constraint you wish to remove.
3. In the **Edit** menu, click **Remove Selected Constraints**.

To remove select constraints on the **Summary** tab, delete the constraint in the **Required** text box and click **Set**.

Export Results

From the **Paths** tab, you can export the path or set grids in a text file.

To save your results to a text file:

1. Click the **Paths** tab.
2. In the **File** menu, click **Export Path Grid** or **Export Set Grid**. This displays the **Save As** dialog box.
3. Choose a destination on your disk, enter a **File Name** and click **Save**.

Commit before you exit

If you wish to save the constraint requirements entered into **Timer**, you must **Commit** your **Timing** results before exiting **Timer**.

To commit your results choose **Commit** from the **File** menu before exiting, or click **Yes** when asked if you would like to commit your results before exiting. **Timer** saves your timing constraints to **Designer's** temporary design database.

Generate reports

The timing report enables you to quickly determine if any timing problems exist in your design. The timing report lists the following information:

- Delay from input I/O to output I/O (longest or shortest, depending on your Preferences).
- Delay from input I/O to internal registers (longest or shortest, depending on your Preferences).
- Delay from internal registers to output I/O (longest or shortest).
- Delays for each clock network (longest or shortest).
- Delays for interaction between clocks networks (longest or shortest).

To generate a timing report:

1. In the **Tool** menu, click **Report Paths**. The **Timing Report** dialog appears. The External Setup-hold Timing Check box and the Slack Threshold text box are explained in the Timer Report dialog box section.
2. Click **Options** to specify more settings for your report. This displays the **Preferences** dialog box.
3. Verify your timing analysis preferences. Timer uses these preferences to generate your report.
 - **Maximum List Size:**
 - Longest/Shortest Path(s)** - Defines the number of paths to display in the report (default is 1)
 - Expanded Path(s) in List** - Defines the number of expanded paths to display in the report (default is 15)
 - **Sort By:**
 - Actual** - Sort paths by actual delays.
 - Slack** - Sort paths by slack delays.This option is available only if you have entered timing constraints.
 - **Case:** specifies your operating conditions, **Best** (0 degrees centigrade), **Typical** (25 degrees centigrade), and **Worst** (70 degrees centigrade).
 - **Break Path at Register** - The default timing paths break at all clock, gate, clear, and preset pins. Uncheck the boxes in Break Path at Register to generate a timing report that displays paths that pass through (and do not "break") at all Clock, Clear, Gate, and Preset pins.

Once you are satisfied with your selections, click **OK** in the **Preferences** dialog box and then click **OK** in the **Timing Report** dialog box. The timing report is displayed in a separate window.

Violations Report

For families that use the pin-to-pin timing model, the Violations report enables you to obtain constraint results sorted by slack. You can now view Max Delay violations as well as Min Delay violations in the report.

Keyword Filters

Keywords

Keywords enable you to filter out any unwanted paths or instances, making it easier to view critical paths in the design and limiting the paths displayed for a particular set. The use of keywords is only supported for pin-to-pin delay families.

Use keywords to create custom sets for Timer's Paths tab screen.

The use of keywords is only supported for pin-to-pin delay families (for an explanation of pin-to-pin and input-to-input delay families, see Delays, PLLs, RAMs, and FIFOs). Use keywords to create custom sets for Timer's **Paths** tab screen. Refer to Adding/Removing sets with keywords for details on how to enter keywords.

Keywords enable you to filter out any unwanted paths or instances, making it easier to view critical paths in the design and limiting the paths displayed for a particular set. Timer uses two types of keywords, first- and second-level.

Levels of Keywords

The first-level keywords enable access to the main objects of the design, such as registers, while the second-level keywords enable access to a sub-list of these main objects. For instance, *\$registers()* is a first-level keyword that enables access to all the registers of the design. This list includes clock pins, data pins, enable pins and, asynchronous pins.

If the *\$registers()* keyword is combined with the second-level keyword *\$datapins()*, the related command is applied only to the data pins of the registers. You can use a second-level keyword only with a first-level; second-level keywords may not be used alone. In Timer, only the first-level keyword *\$registers()* may be combined with the second-level keywords. Use the colon ":" without any spaces to combine first- and second level keywords. Keywords and filters are case insensitive.

Filtering

Filter keywords with brackets []. The filter is a string that is used as an identifier (it may contain wild cards). [] with an empty string is not accepted in the macro language. The user can enter *\$registers()*, *\$registers()[filterString]*, but not *\$registers()[]*.

Functions

Sometimes you may want to locate objects of the design by defining or identifying other objects. For instance, you might want to analyze delays of all the registers driven by a specific primary clock. Functions can help you locate the registers (objects) by defining the primary clock (identifier).

To use functions, the identifier of the object has to be reported between parentheses (). This identifier may contain wild cards and can also be another keyword. For example:

<code>\$registers(clock1)</code>	returns all the registers driven by the primary clock "clock1"
----------------------------------	--

Supported Keywords

Timer supports the keywords listed below.

First-Level Keywords	Second-Level Keywords
<code>\$registers()</code>	<code>\$datapins()</code>
<code>\$inputs()</code>	<code>\$clockpins()</code>
<code>\$outputs()</code>	<code>\$asyncpins()</code>
<code>\$clocks()</code>	<code>\$enablepins()</code>
<code>\$ports</code>	<code>\$outputpins()</code>
	<code>\$inputpins</code>
	<code>\$allpins</code>

First-Level Keywords

Each keyword has two identifiers, a long version and a short version. They both have exactly the same function. The first-level keywords are defined below.

`$registers(ClockName)` OR `$reg(ClockName)`

The keywords above only display the registers (edge-triggered flip-flops and level-sensitive latches) controlled by the clock *ClockName*. If no *ClockName* is specified, this keyword will cause all the registers of the design to be displayed.

`$inputs()` OR `$in()`

This keyword only displays all the primary inputs of the design.

`$outputs()` OR `$out()`

This keyword only displays the primary inputs of the design.

"\$Clocks()" or "\$CK()" only displays the primary clocks of the design.

`$ports(InstanceName)` OR `$po(InstanceName)`

This macro replaces all the primary inputs and outputs of the design.

Second-Level Keywords

While first-level keywords allow access to the main objects of the design, such as registers, second-level keywords give access to a sub-list of these main objects.

Currently, second-level keywords can only be used with the first-level keyword `$registers()`. A first-level keyword is separated from a second-level keyword with the colon ":" character, without any white space.

As with first-level keywords, most second-level keyword have two identifiers, a long version and a short version. Each has the exact same function. In the following examples, it is assumed that the notion of event and pin are implicit.

"\$DataPins()" or "\$dp ()" indicates all the data pins of a register. For example:

<code>\$registers(CLK):\$dp()</code>	displays the data pins of all the registers controlled by CLK
--------------------------------------	---

"\$OutputPins()" or "\$qp()" indicates all the output pins of a register. For example:.

<code>\$registers(CLK):\$qp()</code>	displays the output pins of all the registers controlled by the primary clock CLK
--------------------------------------	---

"\$ClockPins()" or "\$cp()" indicates all the clock pins of a register. For example:.

<code>\$registers(CLK):clockpins()</code>	displays the output pins of all the registers controlled by the primary clock CLK.
---	--

"\$AsyncPins()" or "\$ap()" indicates all the asynchronous pins of a register (preset and clear).

"\$EnablePins()" or "\$ep()" indicates all the enable pins of a register. For example:

<code>\$registers(CLK):\$ep()</code>	displays the enable pins of all the registers controlled by the primary clock CLK.
--------------------------------------	--

`$inputpins()` or `$ip()` indicates all the input pins of a register. For example:.

<code>\$reg(CLK):\$inputpins()</code>	Displays the input pins of all the registers controlled by the CLK
---------------------------------------	--

`$allpins()` indicates all the pins of a register. For example:.

<code>\$registers(CLK):\$allpins()</code>	Displays the pins of all the registers controlled by the CLK
---	--

Second-Level Exceptions

In order to provide more flexibility, the second level keywords can be coupled with exceptions. For instance, if you want to select all the input pins of the registers except the clock pins, you can use the following macro:

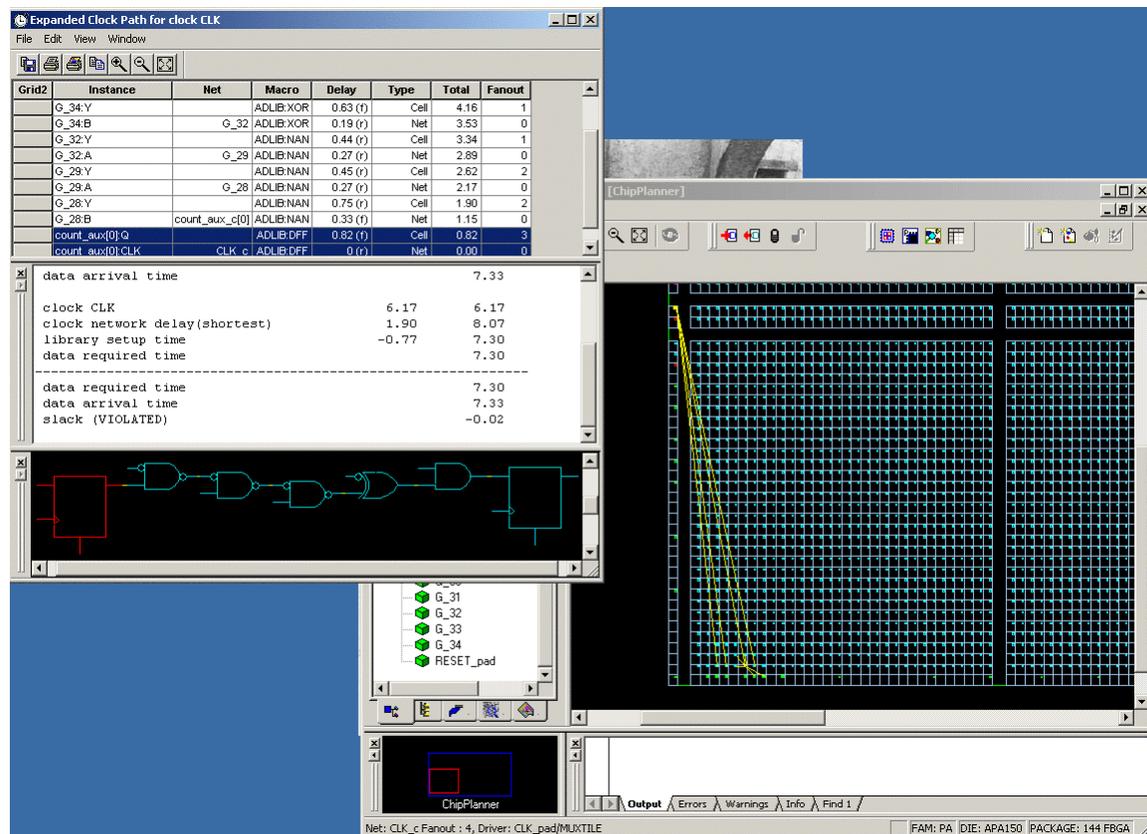
```
$registers(clk):$inputpins(TmacEX_CLOCKPINS)
```

The available exceptions are listed in the following table:

Exception	Result
TmacEX_CLOCKPINS	The clock pins will not be returned from the pins indicated by the 2nd level macro.
TmacEX_DATAPINS	The data pins will not be returned from the pins indicated by the 2nd level macro.
TmacEX_ASYNCpins	The asynchronous pins will not be returned from the pins indicated by the 2nd level macro.
TmacEX_INPUTPINS	The input pins will not be returned from the pins indicated by the 2nd level macro.
TmacEX_ENABLEPINS	The enable pins will not be returned from the pins indicated by the 2nd level macro.
TmacEX_OUTPUTPINS	The output pins will not be returned from the pins indicated by the 2nd level macro.

Using ChipPlanner/ChipEditor with Timer

Use ChipPlanner or ChipEditor and Timer together to view place-and-route of Timer paths.



Timer and ChipPlanner

To view critical paths:

1. Open **Timer** and **ChipEditor/ChipPlanner** from Designer (ChipEditor opens if you are using an ACT1, ACT2, ACT3, DX, MX, SX, SX-A, or eX device; all other devices use ChipPlanner).
2. In Timer, click the **Paths** tab.
3. Select a **Path** set in the path set grid. Paths within that set are displayed in the path grid.
4. Select the path you wish to expand.
5. Expand the path by double-clicking on the path, or in the **Edit** menu, click **Expand Path**. The **Expanded Paths** window opens and displays a single path in

the **Expanded Paths Grid** and a graphical representation of the paths in the **Chart Window**.

6. Select a module or net in the **Expanded Paths** dialog box. The module or net is shown in **ChipEditor** or **ChipPlanner**.

Refer to the online help for more information on how to use the ChipEditor/ChipPlanner tools.

You can add and remove break points in Timer while you use the ChipEditor/ChipPlanner tool.

Timer Glossary of Terms

This glossary defines terms and concepts used in the Timer online help.

clock exception

A terminal in a synchronous network that should be excluded from the specified clock period. The exception can remain undefined (don't care) or can be assigned a unique value in the Path Constraint Editor.

critical path

The path within a design that dictates the fastest time at which an entire design can run. This path runs from the source to a sink node such that if any activity on the path is delayed by an amount t , then the entire circuit function is delayed by time t .

delay constraint

A delay constraint defines a fixed amount of time required for a signal to propagate from all starting terminals to all ending terminals for a network.

destination

An ending point, sink node, for a timing analysis path, often the data input of a synchronous element or pad.

don't care path

A signal path in which the delay is considered to be infinite.

Dynamic Timing Analysis

Dynamic timing analysis (simulation) has been the standard mechanism in verifying design functionality and performance. Both pre-layout and post-layout timing analysis can be performed via the SDF interface. Pre-layout timing analysis provides quick estimates of the designs performance. Post-layout timing simulation on the other hand provides accurate timing information that is appropriate for device or system level simulation.

filter

A set of limitations or options applied to the timing analysis to more specifically target important items of interest.

global Stop

A defined intermediate point in a network that forces all paths through the defined point to be don't care paths regardless of any constraint assignment.

network

Network. A network can consist of 1 or more start terminals and 1 or more end terminals. All signal paths connecting any start terminal to any end terminal are included in the network. Only one delay value can be assigned to each defined network. Networks can be defined implicitly by a common clock (synchronous network) or explicitly by a defined set of terminals. Network and Paths are used interchangeably.

path

An ordered set of elements identifying a logic flow pathway through a circuit. A path may consist of a single net or a grouping of related nets and components. There can be multiple, or parallel paths (consisting of nets and components) between the two pins. When a component is selected as part of a path, both the input pin to the component and the output pin are included in the path. A path stops when it reaches the data input of a synchronous element (flip-flop), clock, or pad. A path usually starts at the output of a synchronous element, clock, or pad.

path delay

The path delay defines the sum of all the individual delays of the nets and the logic macros in the signal path.

path sets

In this manual we refer to groups or categories of paths as "sets." Path sets are displayed on the Paths tab.

signal path

The signal path describes a consecutive sequence of logic and nets, the first net being driven by a start terminal, and the last net driving a macro input pin of the end terminal

slack

The difference between the constraint and the analyzed value, with negative slack indicating the analyzed value is greater than the constrained value.

Standard Delay Format (SDF)

Standard Delay Format is an industry-standard file format used for storing timing data generated by EDA tools. It is often used for simulation.

Static Timing Analysis (STA)

Static timing analysis is an exhaustive and convenient method of ensuring that the design meets its timing requirements. There are functions that are especially easy to analyze with the static approach. Complex functions such as a multiplier are much easier to analyze using the static approach because static analysis offers one hundred percent coverage with minimal effort compared to dynamic timing analysis. In addition, the static approach is faster for highly synchronous designs compared to dynamic timing analysis.

status bar

The area located at the bottom of an application window

Timing delay constraint definitions

The following terminology appears in the description of constraints for Timer.

DTL Terminals

Timing Driven Layout terminals define the starting (or source) and ending (or sink) points for a signal path. They are always I/Os or sequential elements; no intermediate combinatorial element is currently supported as a terminal.

Signal Path

The signal path describes a consecutive sequence of logic and nets, the first net being driven by a start terminal, and the last net driving a macro input pin of the end terminal.

Network

A network can consist of 1 or more start terminals and 1 or more end terminals. All signal paths connecting any start terminal to any end terminal are included in the network. Only one delay value can be assigned to each defined network. Networks can be defined implicitly by a common clock (synchronous network) or explicitly by a defined set of terminals. Network and Paths are used interchangeably.

Path Delay

The path delay defines the sum of all the individual delays of the nets and the logic macros in the signal path.

Delay Constraint

A delay constraint defines a fixed amount of time required for a signal to propagate from all starting terminals to all ending terminals for a network.

Don't Care Path

A signal path in which the delay is considered to be infinite.

Global Stop

A defined intermediate point in a network that forces all paths through the defined point to be don't care paths regardless of any constraint assignment.

Clock Exception

A terminal in a synchronous network that should be excluded from the specified clock period. The exception can remain undefined (don't care) or can be assigned a unique value in the Path Constraint Editor.

Synthesis

Synthesis Overview

Libero IDE works with the following synthesis tools:

- Synplify from Synplicity
- LeonardoSpectrum from Mentor Graphics
- Precision RTL from Mentor Graphics

While LeonardoSpectrum and Precision RTL are not part of the Libero IDE package, they can be integrated to work with Libero IDE. You can also integrate different versions of Synplify. To integrate tools, add them to your project profile.

Synplify

Libero's integrated synthesis tool, Synplify AE from Synplicity, takes your Verilog or VHDL Hardware Description Language source as input and outputs an optimized EDIF and HDL netlist.

Synthesizing your design with Synplify

1. In Libero, right-click the HDL file in the **File Manager**, or the top-level schematic for mixed schematic-HDL designs, in the **Design Hierarchy**, and select **Synthesize**. Synplify starts and loads the appropriate design files, with a few pre-set default values.
2. From Synplify's **Project** menu, click **Implementation Options**.
3. Set your specifications and click **OK**.

4. Click the **RUN** button. Synplify compiles and synthesizes the design into an EDIF, *.edn, file. Your EDIF netlist is then automatically translated by Libero into an HDL netlist. The resulting *.edn and *.vhd files are visible in the File Manager, under Implementation Files

Should any errors appear after you click the Run button, you can edit the file using the Synplify editor. Double-click the file name in the Synplify window showing the loaded design files. Any changes you make are saved to your original design file in Libero.

5. From the **File** menu, click **Exit** to close Synplify. A dialog box asks you if you would like to save any settings that you have made while in Synplify. Click **Yes**.

Integrating Precision RTL

Libero IDE supports Precision RTL from Mentor Graphics.

To integrate Precision RTL with your Libero IDE project:

1. From the **Options** menu, click **Profile**. The Project Profile dialog box appears.
2. Click **Add** and select **Synthesis**. The Add Tool dialog box appears.
3. Enter a name. This is the name that appears in the Project Profile dialog box.
4. Select Precision RTL.
5. Enter the location of Precision RTL and any additional parameters.
6. Click **OK**.
7. Select Precision RTL in the Project Profile dialog box and click **OK**.
8. Double-click **Precision RTL** in the Libero Process window to start Precision RTL.

Starting Precision RTL

Before you can start Precision RTL you must add it to your project profile.

To start Precision RTL to run synthesis:

1. In Libero, right-click the HDL file in the File Manager or the top-level schematic for mixed schematic-HDL designs in the Design Hierarchy, and click **Synthesize**. Precision starts.
2. (Optional) Click **Setup Design** to enter clock frequency, input delays and output delays.
3. (Optional) Click **Constraint** if you want to import a constraint file (*.sdf).
4. Click **Compile**, if you want to compile the design first.
5. If compile runs without error, click **Synthesize** to optimize the design for your target technology. To investigate errors in the log window, click the red error icon next to the error. A HDL Text Editor opens and the part of the HDL text which is the source of the error is automatically highlighted for you to modify. Click **Save** to save the changes you have made to the HDL text. Rerun Synthesis to get a successful run.
6. Click **Synthesize**. Precision RTL runs compile and then synthesizes your design.
7. The synthesized netlist (EDIF format) and is visible under Implementation Files in the Libero File Manager tab.
8. From the **File** menu, click **Exit** to close Precision RTL. A dialog box asks you if you would like to save any settings that you have made while in Precision. Click **Yes** to save the Precision project file (*.psp).

Integrating LeonardoSpectrum

Libero IDE supports LeonardoSpectrum from Mentor Graphics.

To integrate LeonardoSpectrum with your Libero IDE project:

1. From the **Options** menu, click **Profile**. The Project Profile dialog box appears.
2. Click **Add** and select **Synthesis**. The Add Synthesis Tool dialog box appears.
3. Enter a name. This is the name that appears in the Project Profile dialog box.
4. Select LeonardoSpectrum.
5. Enter the location of LeonardoSpectrum and any additional parameters.
6. Click **OK**.
7. Select LeonardoSpectrum in the Project Profile dialog box and click **OK**.
8. Double-click **LeonardoSpectrum** in the Libero Process window to start LeonardoSpectrum.

Synthesizing your design with LeonardoSpectrum

Before you can start Precision RTL you must add it to your project profile.

To synthesizing your design with LeonardoSpectrum:

1. In Libero, right-click the HDL file in the File Manager, or the top-level schematic for mixed schematic-HDL designs, in the Design Hierarchy, and select **Synthesize**. LeonardoSpectrum starts.
2. The Input Box is blank. Hit the Enter key on the keyboard and all the design files are loaded into the Input Box.
3. From **Tools**, click **Options**.
4. Unselect Automatically save and restore Current Work Directory option.
5. Enter the Clock Frequency if you want to constrain the design.
6. Use the slide bar to select the level of Optimize Effort.

7. Click **Run Flow**. Information about errors can be found by clicking the red error icon next to the error. The HDL Text Editor opens and the part of the HDL text which is the source of the error is automatically highlighted for you to modify.
8. Click **Save** to save the changes you have made to the HDL text. Rerun Synthesis to get a successful run. The synthesized netlist is in EDIF format and is visible under Implementation Files in the Libero File Manager tab.
9. From the **File** menu, click **Exit** to close LeonardoSpectrum. A dialog box asks you if you would like to save any settings that you have made while in LeonardoSpectrum. Click **Yes** to save the LeonardoSpectrum project file (*.lsp).

Integration issues

Some workarounds are required when using LeonardoSpectrum with Libero IDE.

- **LeonardoSpectrum starts with empty windows:** When you first open LeonardoSpectrum from Libero, the Input window is blank and the output file is not specified. Also, the wrong family appears in the Technology window. To fix this, simply place your cursor in the transcript window and press **Enter**. All windows are updated.
- **LeonardoSpectrum log files are misplaced:** From the **Tools** menu, click **Options**, and **Session Settings**. Unselect Automatically save and restore current working directory. This box must be un-selected in order for your log files to be passed back to Libero properly.

Testbench Creation

WaveFormer Lite

WaveFormer Lite is a special version of WaveFormer Pro that can generate VHDL and Verilog stimulus-based testbenches for Libero IDE. WaveFormer Lite fits perfectly into Libero's design environment, automatically extracting signal information from your HDL design files and producing HDL testbench code that can be used for VHDL or Verilog simulation.

WaveFormer Lite generates VHDL and Verilog testbenches from drawn waveforms.

WaveFormer Lite can generate the following:

- VHDL transport testbench (*.vhd) that uses assignment statements
- VHDL wait testbench (*.vhd) that uses wait statements
- Verilog (*.v) file with Verilog stimulus statements

Note:

- WaveFormer Lite comes with its own online help. After starting WaveFormer Lite, click the **Help** menu.

Creating your testbench with WaveFormer Lite

WaveFormer Lite generates VHDL and Verilog testbenches from drawn waveforms.

Create your testbench after you are done creating your design and wish to perform simulation.

There are five basic steps for creating testbenches using WaveFormer Lite and Libero IDE. These steps are described in detail in the following sections.

To create a testbench using WaveFormer Lite:

1. Double-click **WaveFormer Lite** in the **Process Window**. WaveFormer Lite starts and your signal information is automatically imported.

2. Using WaveFormer Lite, draw the waveforms to describe the testbench.
3. (Optional) Add VHDL Libraries and Use Clauses for VHDL export. These libraries or packages can be included using the VHDL Libraries and Use Clauses dialog. From the **Options** menu, click the **VHDL Libraries and Use Clauses** menu item to open this dialog.
4. From the **Export** menu, click **Export Timing Diagram** and choose the type of file to generate. You can generate a testbench with a top-level module that automatically hooks up the model under test to the testbench, or you can generate just a testbench model. Below is a detailed description of the two methods:
 - To generate a Top-Level Model and a Testbench model choose one of the "top-level" scripts from the save as type drop-down list box. The top-level module will instantiate the model under test and hook it up to the testbench. For this script to work the top-level module needs to be defined in the project. For Wave-Former Lite customers, the Actel Software should automatically set this option. Below is a list of top-level scripts:
 - VHDL Wait with Top Level TestBench (*.vhd)s
 - VHDL Transport with Top Level TestBench (*.vhd)s
 - Verilog with Top Level TestBench (*.v)s
 - To generate a plain testbench model (which does not instantiate your model under test) then choose one of the VHDL or Verilog scripts. To simulate with the testbench model, you will need to write a top-level model that instantiates the testbench model and the model under test. This is the method used by Wave-Former Pro customers. Below is a list of VHDL and Verilog testbench generation scripts:
 - VHDL Wait (*.vhd)s
 - VHDL Transport (*.vhd)s
 - Verilog
5. From the **File** menu, click **Exit**.

Note:

- If you added extra signals to the testbench and do not want to export those signals, then double click the signal's names to open the Signals Properties dialog and uncheck the Export check box.

Simulation

ModelSim AE

ModelSim Actel Edition (AE) is a custom edition of ModelSim PE that is integrated into Libero's design environment. ModelSim for Actel is an OEM edition of Model Technology Incorporated's (MTI) tools. ModelSim for Actel supports VHDL or Verilog, but it can only simulate one language at a time. It only works with Actel libraries and is supported by Actel.

Other editions of ModelSim are supported by Libero. To use other editions of ModelSim with Libero, simply do not install ModelSim PE from the Libero CD.

Note:

- ModelSim for Actel comes with its own online help and documentation. After starting ModelSim, click the Help menu.

Setting your simulation options

You can set a variety of simulation options for your project.

To set your simulation options:

1. From the **Options** menu, click **Project Settings**.
2. Click **Simulation**.
3. Select your options and click **OK**.
 - **Use automatic do file:** Select if you do not want Libero to initialize ModelSim.
 - **User defined Do file:** Enter the do file name or click the browse button.
 - **Compile VHDL Package files:** Select to compile VHDL package files using ModelSim AE.

- **Include Do file:** Select to execute the wave.do or other specified Do file. Use the wave.do file to customize the ModelSim Waveform window display settings.
- **Simulation Run Time:** Specify how long the simulation should run in ns. If the value is 0, or if the field is empty, there won't be a run command included in the run.do file.
- **Testbench entity name:** Specify the name of your testbench entity name. Default is "testbench," the value used by WaveFormer Lite.
- **Top Level instance name in the testbench:** Default is <top_0>", the value used by WaveFormer Lite. Libero replaces <top> by the actual top level macro when ModelSim is run.
- **Vsim Command Type:**Select Min, Typical (Typ), or Max
- **Resolution:** The default is family specific, but you can customize it to fit your needs.

Family	Default Resolution
ACT1, ACT2, ACT3	1 ns
MX	1 ns
DX	1 ns
SX, SX-A	1 ns
eX	1 ns
Axcelerator	1 ps
ProASIC	1 ps
ProASIC ^{PLUS}	1 ps

- **Vsim additional options:** Text entered in this field is added to the vsim command.
- **Default:** Restores factory settings.

Selecting a stimulus file for simulation

Before running simulation, you must associate a testbench. If you attempt to run simulation without an associated testbench, Libero IDE asks you to associate a testbench or open *ModelSim* without a testbench.

To associate a stimulus:

1. Run simulation or right-click the top level module in the Design Hierarchy Menu and click **Select a Stimulus File** from the right-click menu. The Select Stimulus dialog box appears.
2. Associate your testbench(es):

In the Select Stimulus dialog box, all the stimulus files in the current Libero project appear in the left *Stimulus Files in the Project* list box. Files already associated with the block appear in the *Associated Files* list box.

In most cases you will only have one testbench associated with your block.

However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the Associated Files dialog box.

To add a testbench: Select the testbench you want to associate with the block in the Stimulus Files in the Project list box and click **Add** to add it to the Associated Files list.

To remove a testbench: To remove or change the file(s) in the Associated Files list box, select the file(s) and click **Remove**.

To order testbenches: Use the up and down arrows to define the order you want the testbenches compiled. The top level entity should be in the bottom of the list.

4. When you are satisfied with the Associated File(s) list, click **OK**. A check mark appears next to WaveFormer Lite in the Process window to let you know that a testbench has been associated with the block.

Performing functional simulation

1. Create your testbench.
2. Right-click the top level module in the **Design Hierarchy** window.
3. Click **Select a Stimulus File** from the right-click menu.

In the Select Stimulus dialog box, all the stimulus files in the current Libero project appear in the left *Stimulus Files in the Project* list box. Files already associated with the block appear in the *Associated Files* list box.

In most cases you will only have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the *Associated Files* dialog box.

To add a testbench: Select the testbench you want to associate with the block in the *Stimulus Files in the Project* list box and click **Add** to add it to the Associated Files list.

To remove a testbench: To remove or change the file(s) in the Associated Files list box, select the file(s) and click **Remove**.

To order testbenches: Use the up and down arrows to define the order you want the testbenches compiled.

4. When you are satisfied with the Associated File(s) list, click **OK**. A check mark appears next to WaveFormer Lite in the Process window to let you know that a testbench has been associated with the block.

5. Start *ModelSim* AE by doing one of the following:
 - Right-click the top level module in the Design Hierarchy window and select **Run Pre-Synthesis Simulation** or **Run Post-Synthesis Simulation**.
 - Double-click *ModelSim* Simulation in the **Process Window**.
ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1uS and the Wave window opens to display the simulation results.
6. Scroll in the Wave window to verify that the logic of your design functions as intended. Use the zoom buttons to zoom in and out as necessary.
7. From the **File** menu, click **Quit**.

Performing timing simulation

The steps for performing functional and timing simulation are nearly identical. Functional simulation is performed before place-and-route and simulates only the functionality of the logic in the design. Timing simulation is performed after the design has gone through place-and-route and uses timing information based on the delays in the placed and routed designs.

Timing simulation includes much more detailed timing information for the targeted device. Timing simulation requires a testbench.

To perform timing simulation:

1. If you have not done so, back-annotate your design and create your testbench.
2. Right-click the top level module in the **Design Hierarchy Menu**.
3. Click **Select a Stimulus File** from the right-click menu.

In the Select Stimulus dialog box, all the stimulus files in the current Libero project appear in the left *Stimulus Files in the Project* list box. Files already associated with the block appear in the *Associated Files* list box.

In most cases you will only have one testbench associated with your block.

However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the *Associated Files* dialog box.

To add a testbench: Select the testbench you want to associate with the block in the *Stimulus Files in the Project* list box and click **Add** to add it to the *Associated Files* list.

To remove a testbench: To remove or change the file(s) in the *Associated Files* list box, select the file(s) and click **Remove**.

To order testbenches: Use the up and down arrows to define the order you want the testbenches compiled.

4. When you are satisfied with the *Associated File(s)* list, click **OK**. A check mark appears next to *WaveFormer Lite* in the *Process* window to let you know that a testbench has been associated with the block.
5. Double-click **ModelSim Simulation** in the *Process* window. The *ModelSim* simulator starts and compiles the source files. When the compilation completes, the simulator runs for 1 uS and a *Wave* window opens to display the simulation results.
6. Scroll in the *Wave* window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays. If you didn't create a testbench with *WaveFormer Lite*, you might get error messages with the *vsim* command if the instance names of your testbench don't follow the same conventions as *WaveFormer Lite*. Ignore the error message and type and the correct *vsim* command.
7. When you are done, from the **File** menu, click **Quit**.

Device Programming

Generating Programming Files

Once you have completed your design, and you are satisfied with the back-annotated timing simulation, create your programming file. Depending upon your device family, you need to generate a Fuse, Bitstream or STAPL programming file.

Programmer	Antifuse Programming File	Flash Programming File
Flash Pro	N/A	.stp
Silicon Sculptor I	.afm (Non-Axcelerator families)	.bit
Silicon Sculptor II	.afm	.bit .stp (Windows only)

Starting Silicon Sculptor from Libero IDE

Before starting Silicon Sculptor, generate your programming file.

To start the programming tool software:

1. Right-click the design root file in the Design Hierarchy window.
2. Click **Run Silicon Sculptor**. Refer to the Silicon Sculptor User's Guide for information on using the programming tool.

FlashLock

Actel's ProASIC and ProASIC^{PLUS} devices contain FlashLock circuitry to lock the device by disabling the programming and readback capabilities after programming. Care has been taken to make the locking circuitry very difficult to defeat through electronic or direct physical attack.

FlashLock has three security options: No Lock, Permanent Lock, and Keyed Lock.

No Lock

Creates a programming file which does not secure your device.

Permanent Lock

The permanent lock makes your device one time programmable. It cannot be unlocked by you or anyone else.

Keyed Lock

Within each ProASIC and ProASIC^{PLUS} device, there is a multi-bit security key user key. The number of bits depends on the size of the device. The tables below show the key size of different ProASIC and ProASIC^{PLUS} devices, respectively. Once secured, read permission and write permission can only be enabled by providing the correct user key to first unlock the device. The maximum security key for the device is shown in the dialog box.

Key Size of ProASIC Devices

Device	Key Size (bits)	Key Size (Hex)
A500K050	51 Bits	13
A500K130	51 Bits	13
A500K180	51 Bits	13
A500K270	51 Bits	13

Key Size of ProASIC^{PLUS} Devices

Device	Key Size (bits)	Key Size (Hex)
APA075	79 Bits	20
APA150	79 Bits	20
APA300	79 Bits	20
APA450	119 Bits	30
APA600	167 Bits	42
APA750	191 Bits	48
APA1000	263 Bits	66

Programming the Security Bit

Two device programmers, Silicon Sculptor and Flash Pro, are available for ProASIC and ProASIC^{PLUS} devices. If the programming file contains the security key, by default the Silicon Sculptor and Flash Pro programming software automatically enables the "secure" option and programs the security key. You can turn this off, should you decide not to program using the security key.

Please refer to the application note "Implementation of Security in Actel's ProASIC and ProASIC^{PLUS} Flash-Based FPGAs" for more details.

Generating Bitstream and STAPL files

Bitstream allows you to generate a bitstream or STAPL file for ProASIC and ProASIC^{PLUS} devices. Please consult the Program Files table to find out which file type you should choose.

To generate a bitstream or STAPL file:

1. In the **Tools** menu, click **Bitstream** or click the Bitstream button in the Design Flow window.
2. Select **Bitstream** or **STAPL** from the File Type drop-down list box.
3. **FlashLock**. Select one of the following options:
 - **No Locking**: Creates a programming file which does not secure your device.
 - **Use Keyed Lock**: Creates a programming file which secures your device with a FlashLock key. The maximum security key for the device is shown in the dialog box. The maximum security key for the device is shown in the dialog box.
 - **Use Permanent Lock**: Creates a one-time programmable device.
4. Click **OK**. Designer validates the security key and alerts you to any concerns.

Note: The bitstream file header contains the security key.

Generating a Fuse file

Fuse allows you to generate a programming file for your Actel Antifuse devices. Fuse files work with Actel's Silicon Sculptor programmers. (For Axcelerator families, you must use the Silicon Sculptor II programmer.)

To generate a fuse programming file:

1. In the **Tools** menu, click **Fuse** or click the **Fuse** button in the Design Flow window.
 - **File Type.** Select the appropriate file type in the File Type pull-down menu. Select "AFM-APS2" if you are using Silicon Sculptor programmer.
 - **Silicon Signature (Optional):** Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are "0" through "9," and "a" through "f."
 - **Output filename:** Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box. Click **Browse** to change the directory. Do not add a file extension or suffix to the file name. The Designer software automatically adds the extension to the programming file name when you specify the programming format.
 - **Generate Probe File Also:** This option automatically generates a .prb file for use with Silicon Explorer
 - **Disable clamping diode for unused I/O pins:** (SX-A and eX families). Check box to disable clamping diode.
 - **Use the JTAG Reset Pull-up Resistor:** (Axcelerator family) Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).

- **Use the Global Set Fuse:** (Axcelerator family) Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option is used, all flip-flops are set to '1' at power up.
2. Click **OK** when finished to save the file.

Generating prototype files

When designing for RTAX-S, you can use the Axcelerator family of devices for prototyping. Please refer to the application note, Prototyping RTAX-S Using Axcelerator Devices for more information.

To generate prototype files:

1. From the **Tools** menu, click **Generate Prototype**. In the Generate Prototype Files dialog box, make the following selections:
 - **Silicon Signature (Optional).** Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are “0” through “9,” and “a” through “f.”
 - **Output filename.** Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box. Click Browse to change the directory. Do not add a file extension or suffix to the file name. The Designer software automatically adds the extension to the programming file name when you specify the programming format.
 - **Generate Probe File Also.** This option automatically generates a .prb file for use with Silicon Explorer

- **Use the JTAG Reset Pull-up Resistor:** Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).
 - **Use the Global Set Fuse:** Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option is used, all flip-flops are set to '1' at power up.
2. Click **OK**. The AFM file is generated.

Contacting Actel

Actel Headquarters

Actel Corporation is a supplier of innovative programmable logic solutions, including field-programmable gate arrays (FPGAs) based on antifuse and flash technologies, high-performance intellectual property (IP) cores, software development tools and design services targeted for the high-speed communications, application-specific integrated circuit (ASIC) replacement, and radiation-tolerant markets.

Address:	955 East Arques Avenue Sunnyvale, CA 94086
Phone:	(888) 99-ACTEL

Technical Support

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M. Pacific Time, Monday through Friday.

Visit Tech Support Online

For 24-hour support resources, visit Actel Technical Support at <http://www.actel.com/custsup/search.html>.

Contacting Technical Support

Contact us with your technical questions via e-mail or by phone. Also, if you have design problems, you can e-mail your design files to receive assistance. When sending your request to us, please be sure to include your full name, company name, and telephone number.

E-mail:	tech@actel.com
Telephone (In U.S.):	(408) 522-4460 (800) 262-1060
Telephone (Outside the US):	Contact a local sales office

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization. For technical issues, contact Technical Support.

From	Call
Northeast and North Central U.S.A.	(408) 522-4480
Southeast and Southwest U.S.A.	(408) 522-4480
South Central U.S.A.	(408) 522-4434
Northwest U.S.A.	(408) 522-4434
Canada	(408) 522-4480
Europe	(408) 522-4252 or +44 (0) 1276 401500
Japan	(408) 522-4743
From the rest of the world	(408) 522-4743

Sales

For a list of Actel Sales Offices, please see the Actel web site <http://www.actel.com/contact/offices/offices.html>.

Documentation Feedback

Actel Corporation strives to produce the highest quality online Help and printed documentation. We want to help you learn about our products. We welcome your feedback. Please send your comments to documentation@actel.com.

Index

.lok 109

A

Actel 293

ADB 112

Adding applications 110

ADL 129

AFL 129

AFM 129

Applications 110

Associate 286

Audit settings 118

Auditing 118

Auxiliary files 96, 118

B

BIT 129

Bitstream 289, 290

BSD 129

C

ChipEditor 42

ChipPlanner 48, 50

COB 129

constraint file 95

constraints 66, 95, 100

crash 109

CRT 129

D

DCF 95, 129

Device Support 12

DIO 129

Directory 111

Documentation 297

E

EDN 129

Empty 48

Exclusive 48

Exporting 66, 129

Exporting files 129

F

Feedback 297

Files 12, 13, 14, 15, 16, 21, 24, 25,
27, 32, 33, 283, 286, 287, 289,
290, 291

Flash Layout 123

Flash Pro 289

FlashLock 289, 290

FlashPro 19

Flip-Flop 132

Floorplanning 47, 48, 50

FUS 129

Fuse 289, 291

G

GCF 66, 95, 96, 100, 118, 129

Guidelines 31

H

HDL Editor 7, 25, 27

I

Importing 15, 66, 96, 100, 116, 118,
120

Inclusive 48

Internet 111, 112

K

Keyed lock 289

L

LeonardoSpectrum 281

Liberio Gold 5, 7

Liberio Platinum 5

-
- Libero Platinum Evaluation* 5
Libero Silver 5
Literature 297
LOC 129
LOG 129
logic assignment 44
- M**
- macro* 42
Menu 12, 13, 15, 16, 21, 24, 25, 27, 33, 287, 290
ModelSim 286
- N**
- naming* 31
New Project 12
New tools 110
New version 111
- O**
- Opening* 111
ordering applications 110
- P**
- PDC* 66, 79, 89, 90, 92, 96, 118, 129, 168
PDF 113
Permanent lock 289
PIN 129
PinEditor 35, 44
place and route 123
Placement 42
Port names 31
PRB 129
Precision RTL 280
Preferences 111, 112, 113
ProASIC 66, 100
Programming 19, 289, 291, 292
Projects 12, 13, 14, 15, 16, 23, 24, 25, 27, 32, 33, 283
Prototype 292
Proxy 112
- R**
- Regions* 47, 48, 50
Reports 132
RTAX-S 292
- S**
- SAIF* 129
Sales 294
Saving 111
Schematic 31, 32, 33
Sculptor 289, 291
SDC 95, 96, 97, 99, 118, 120
SDF 129
Security 289
Security Key 290
SEG 129
Simulation 287
simulation options 18, 285
Software update 111
Source files 116
STAPL 289
Starting 35
Starting applications 110
STF 129
Stimulus 286
Synthesis 280, 281
- T**
- TCL* 79, 89, 90, 92, 129, 168, 169, 177, 179, 180, 197
Technical Support 293
Testbench 7, 14, 283, 286, 287
Timing 287

Timing driven layout 123

V

Tools menu 110

VDC 129

Troubleshooting 293

Version Checking 21, 111

U

W

UNIX 113

WaveFormer Lite 286

Updates 111