

Actel® Tools
Designer User's Guide
R1-2003



Actel® Corporation, Sunnyvale, CA 94086

© 2002 Actel Corporation. All rights reserved.

Part Number: 5029122-4

Release: December 2002

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, Action Logic, Activator, and Actionprobe are registered trademarks of Actel Corporation.

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

Sun Workstations and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc.

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

UNIX is a registered trademark of UNIX System Laboratories, Inc.

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

Table of Contents

Introduction	xi
Document Organization	xi
Document Assumptions	xii
Platform Support	xii
Your Comments	xii
Online Help	xiii
1 Designer: Getting Started	11
Starting Designer	11
Initiating the Designer Session	13
Designer.	16
Designer's Menu Commands.	17
Toolbar	20
Log Window	21
Status Bar	22
2 Using Designer	23
Starting and Initiating a Design Session	24
Importing Source Files	27
Importing Auxiliary Files	30
Importing PDC Files (Axcelerator family only)	33
Importing SDC Files	35
Auditing Files	37
Device Selection Wizard	39
Compiling a Design.	44
Compile Options	45
User Tools	47
Place-and-Route Variables (Non-ProASIC and ProASIC PLUS Families)	
49	
Layout.	50
Layout Options	53
Layout Failures	57

Back-Annotation58
Generating Programming Files60
Changing Design Name and Family65
Changing Design Information66
Exporting Files67
Generating Reports72
Setting Designer Preferences83
Starting other Applications from Designer86
Saving a Design87
License Details88
Ending the Designer Session88
3 Scripting89
Tcl Overview89
Tcl Extension Commands Added by Designer95
PDC Commands	100
SDC Commands	104
Tcl Commands for Timer	107
Tcl Commands for PinEdit	111
Running Scripts from the Command Line	113
Running Scripts within Designer	114
Recording Scripts	114
Example Scripts	116
A Constraints in ProASIC and ProASIC PLUS Devices	121
Types of Constraints	121
ProASIC Timing Constraints	121
Highlevel Timing Constraints	122
Timing Constraints	125
Global Resource Constraints	127
Netlist Optimization Constraints	132
Constraint Quick Reference	141
Constraint File Syntax Summary	142

B	Setting Up a Printer in UNIX	149
C	Product Support	155
	Actel U.S. Toll-Free Line	155
	Customer Service	155
	Actel Customer Technical Support Center	155
	Guru Automated Technical Support	156
	Web Site.	156
	Contacting the Customer Technical Support Center.	156
	Worldwide Sales Offices	158
	Index	159

List of Figures

Designer Software	12
Setup Design Dialog Box	13
Open Design Dialog Box	14
Designer, Design Session Initiated (PC Version)	16
Designer Toolbar	20
Design Tools Toolbar	20
Log Window All Anti-fuse Families	21
Starting Designer	24
Setup Design Dialog Box	25
Open Design Dialog Box	26
Import Source Files Dialog Box	28
Add Source File Dialog Box	29
Import Source Files Dialog Box with EDIF File Added	29
Import Auxiliary Files Dialog Box	31
Add Auxiliary Files	31
File Added to the Import Auxiliary Files Dialog	32
Importing a PDC File Dialog Box	34
Selecting the PDC Files	34
PDC File Added to the Import Auxiliary Files Dialog	35
Import Auxiliary Files Dialog Box	36
Adding an SDC File	36
Audited File is Out of Date Dialog Box	37
Update Audit Source Files Dialog Box	38
Device Selection Dialog Box	39
Device Selection Wizard, Device Variations (Screen Varies Depending Upon Device)	40
Device Selection Wizard - Variations for the Axcelerator Family	41
Device Selection Wizard, Operating Conditions	42
Compile Options Dialog Box, Axcelerator Family	45
Compile Options Dialog Box, SX Family	45
Set Variable Dialog Box	49
Axcelerator Layout Options Dialog Box	51

List of Figures

Other Anti-fuse Families Layout Options Dialog Box52
ProASIC and ProASIC ^{PLUS} Options Dialog Box52
Advanced Layout Options (SX, SX-A, and eX)53
BackAnnotate Dialog Box59
Generate Fuse File Dialog Box61
Bitstream Generation Dialog Box64
Setup Design Dialog Box65
Bitstream Export Options Dialog Box68
Exporting Your PDC File69
Export Timing Files Dialog Box70
Timing Preferences Dialog Box71
Report Type Dialog Box72
Timing Report Dialog Box73
Timing Preferences Dialog Box74
Timing Report76
Pin Report Dialog Box77
Pin Report78
Flip-Flop Report Dialog Box79
Flip-Flop Extended Report79
Flip-Flop Summary Report80
Power Report Dialog Box81
Power Preferences Dialog Box82
Directory Preferences Dialog Box83
Internet Tab (in Preferences Dialog Box)84
File Association Tab85
Customize Dialog Box86
License Details Dialog Box88
Execute Script Dialog Box114
Export Tcl Script Files Dialog Box115
Script Export Options Dialog Box.115
Global Resource Promotion Scheme128
Pad Locations136
Print Dialog Box149

Printer Setup Dialog Box 149
Printer Installation Dialog Box 150
Add Printer Dialog Box 150

List of Figures

Introduction

The Designer User's Guide contains an overview of Actel's Designer software and the design implementation process.

Document Organization

This guide provides detailed cross-platform information about Designer. Use it as a reference in your everyday work.

Step-by-step instructions for using Designer on Windows and Unix workstations are included in this guide. Any platform differences in procedures and commands are noted in the text.

The Using Designer Guide is divided into the following chapters:

Chapter 1 - Getting Started with Designer explains how to invoke and initiate a Designer session and describes Designer's interface, toolbars, and menu commands.

Chapter 2 - Using Designer describes how to use the Designer software to optimize and implement designs to program Actel devices.

Chapter 3 - Scripting contains information on using Tcl inside Designer. A list of all Designer Tcl extension commands is provided.

Appendix A - Using Constraints in ProASIC and ProASIC^{PLUS} Devices explains how to create a ProASIC constraint file (.gcf), which can then be imported into Designer.

Appendix B - Setting up a Printer on UNIX contains directions on how to set your printer up to work with Designer.

Appendix C - Product Support describes our support services.

Appendix D - Revision History contains information on changes made to this guide.

Document Assumptions

The information in this manual is based on the following assumptions:

1. You have installed the Designer Series software.
2. You are familiar with UNIX workstations and UNIX operating systems, or with PCs and Windows operating environments.
3. You are familiar with FPGA architecture and FPGA design software.

Platform Support

Supported Platforms include:

PC

- WinNT 4.0 SP6
- Win2000 SP1
- Win98 2nd Edition
- Windows XP

HP

- HP-UX 11.0

Solaris

- Solaris 7
- Solaris 8

Your Comments

Actel Corporation strives to produce the highest quality online help and printed documentation. We want to help you learn about our products and get your work done quickly. We welcome your feedback about this guide and our online help. Please send your comments to **docs@actel.com**.

Online Help

Designer comes with online help. Online help specific to each Actel software tool is available.

Designer: Getting Started

This chapter will familiarize you with Designer's graphical user interface and associated menu commands. For more information on how to use Designer, refer to "Using Designer" on page 23.

Starting Designer

To start Designer:

1. On a PC

Choose Designer from the Designer group in the Programs menu under the Start menu.

On Unix

Type the following command at the prompt:

<location of Actel software>/bin/designer

Designer opens. If this is your first time using Designer, you are asked if you would like to associate the *.adb file type with Designer. You are also asked if you would like to check for a software update. After making your

selections, Designer opens and is ready for you to begin, as shown in Figure 1-1.

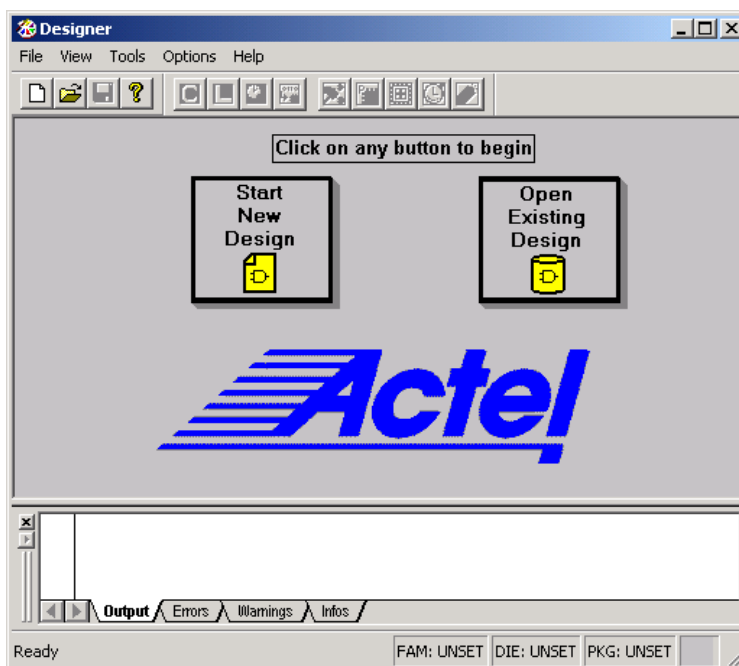


Figure 1-1. Designer Software

Initiating the Designer Session

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

Starting a New Design

1. **Click the *Start New Design* button in the main window, or in the File menu, click *New*.** This displays the Setup Design dialog box, as shown in Figure 1-2.

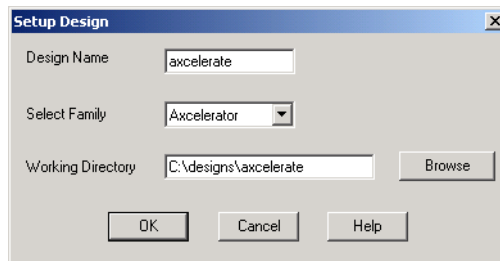


Figure 1-2. Setup Design Dialog Box

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 product Family from the drop down menu list.
- Specify a working directory.

3. **Click *OK*.**

Designer's custom design flow window appears (as shown in Figure 1-4 on page 20). All Designer's tools and commands are activated.

Opening an Existing Design

Designer can open designs that have previously been saved, including designs from previous versions of Designer (refer to the next section, “Opening Designs Created in Previous Versions of Designer”).

To open an existing design:

1. **Click the *Open Existing Design* button or in the File menu, click *Open*.** This displays the Open dialog box (Figure 1-3).

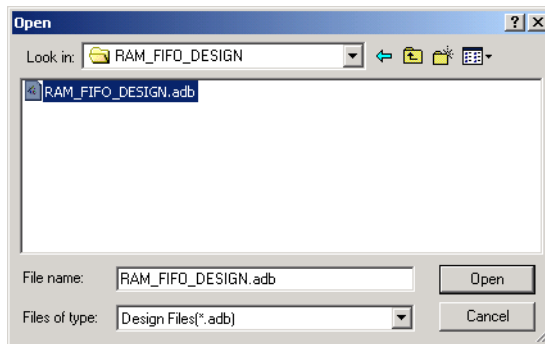


Figure 1-3. Open Design Dialog Box

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*.**

Note: 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.

Designer’s custom design flow window appears (as shown in Figure 1-4 on page 20) and all tools commands are activated.

**Opening
Designs
Created in
Previous
Versions of
Designer**

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

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

All existing die, package, pin assignment, 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.

Designer

Once you have initiated the design session, Designer displays a design flow specific to the family selected in the Setup Design Dialog Box.

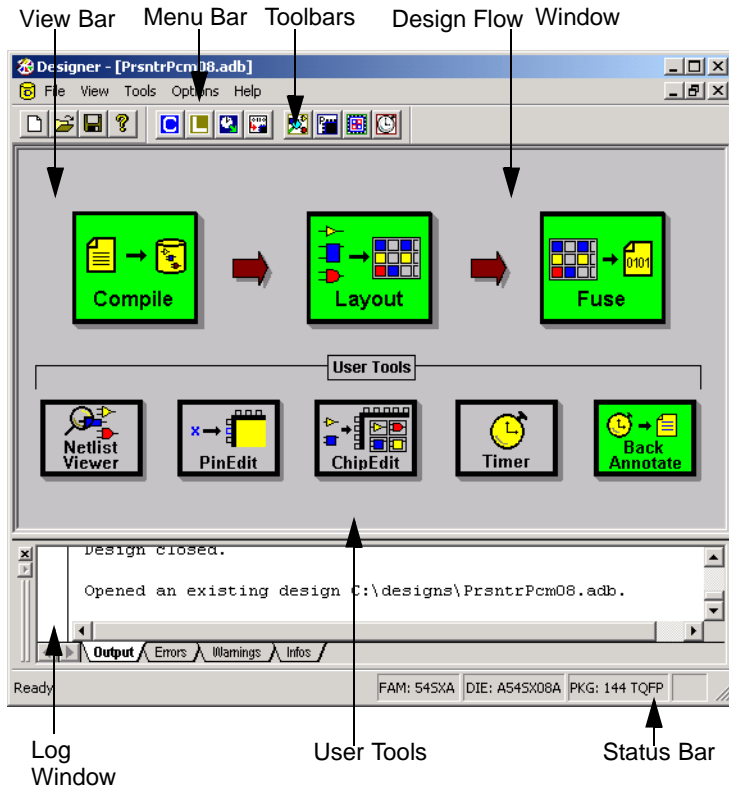


Figure 1-4. Designer, Design Session Initiated (PC Version)

The Design Flow window guides you through the development process, filling in completed steps. The message window displays status and error messages.

Designer's Menu Commands

Menu commands are different between the PC and UNIX platforms. Differences are noted below. 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 Unix versions

File Menu

New: Creates a new design.

Open: Opens an existing design, an *.adb file.

Close: Closes the design.

Save: Saves the design.

Save As: Saves the design with a new name.

Execute Script: Executes a batch script.

Import Source Files: Imports netlist and constraint files.

Import Auxiliary Files: Add, modify or delete associated criticality, SDC, PIN, SAIF, Physical Design Constraint Files (PDC), DCF, and VCD files.

Audit Settings: Changes audit settings for imported source files.

Export: Netlist: Exports the design netlist.

Export: Auxiliary Files: Exports pin, constraint, and BSDL, and placement files.

Export: Fuse: Exports file required to program an antifuse device.

Export: Bitstream: Exports file required to program a ProASIC or ProASIC^{PLUS} device.

Export: Timing Files: Exports timing data for Backannotation.

Export: Script Files: Exports a script file.

Export: Log Files: Exports a log of the current session.

Preferences: Sets design, internet and proxy preferences.

Recent Files: Lists recent files.

Exit: Exits Designer.

View Menu

Toolbar: Displays or hides the Toolbar.

Design Tools: Displays or hides the Design Tools toolbar.

Log Window: Displays or hides the log window.

Status Bar: Displays or hides the Status Bar.

Tools Menu

Setup Design: Selects design name, family, and working directory.

Device Selection: Defines die, package, and other parameters.

Compile: Compiles the loaded design.

Layout: Runs layout on the loaded design.

Back-Annotate: Generates delay data.

Fuse: Generates fuse data for antifuse devices.

Bitstream: Generates a bitstream file for ProASIC and ProASIC^{PLUS} devices.

PinEdit: Starts PinEdit.

ChipEdit: Starts ChipEdit.

Netlist Viewer: Starts the Netlist Viewer tool.

Timer: Starts Timer for static timing analysis.

SmartPower: Starts the power analysis tool.

Reports: Generates status, timing, pin flip-flop, and timing violation reports.

Software Update: Checks Actel website for latest software updates.

Customize: Adds custom macros to Designer's tools menu (PC Only).

Options Menu

Netlist Import: EDIF Naming: Sets default EDIF import options.

Netlist Import: ADL Naming: Sets default ADL import naming style.

Compile: Sets compile options specific to each family.

Get Variable: Displays a selected variable's value.

Set Variable: Sets a selected variable's value.

Clear Log: Clears the session log.

Window (PC Only)

Arrange Icons: Arrange icons at the bottom of the Design window.

Cascade: Arrange windows in the Design window so they overlap.

Tile Horizontally: Arrange windows in the Design window as non-overlapping tiles.

Tile Vertically: Arrange windows in the Design window as non-overlapping tiles.

Close Design: Closes the current design.

Help Menu

Help Topics: Lists help topics.

Reference Manual: Lists available online manuals.

Actel on the Web: Starts your internet browser and opens the Actel website or the Actel Product Support Portal.

License Details: Displays information about your license.

About Designer: Displays program information and current version.

Toolbar

Frequently used commands are available from Designer’s toolbars. Use the View menu to display or hide these toolbars.

If you position the mouse pointer over a toolbar button, a short description (called a tool tip) appears next to the button and a longer description appears in the status bar at the bottom of the main window.

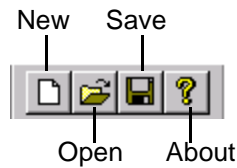


Figure 1-5. Designer Toolbar

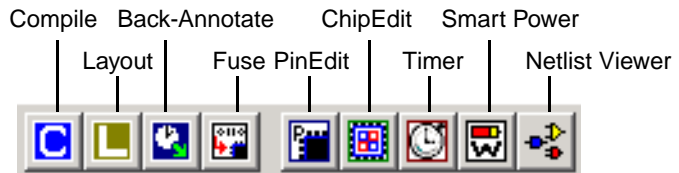





Figure 1-6. Design Tools Toolbar

Log Window

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:

Table A-1.

Type	Symbol	Color
Error		Red
Warning		Blue
Information		Black

While the Output tab displays everything, you can filter for errors, warnings, or informational messages by clicking the other tabs. The views within the error, warnings, and infos displays are reset when a new command is executed or a new design is opened. To see a complete history of your design session(s), click the output tab.

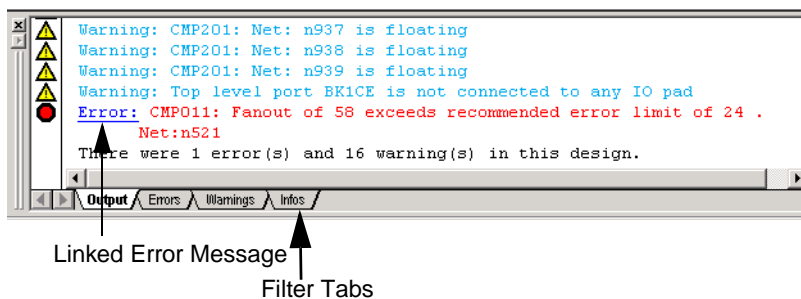


Figure 1-7. Log Window All Anti-fuse Families

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

Status Bar

As you roll your mouse over commands, tool tips appear in the left part of the status bar. Family, die, and package information always appear in the left corner of the status bar.

Using Designer

This chapter contains information on how to use Designer. For an overview of Designer's interface and commands, please refer to “[Designer: Getting Started](#)” on page 15. This chapter covers the general design flow process for implementing your design, including:

- [Starting and Initiating a Design Session on page 24](#)
- [Importing Source Files on page 27](#)
- [Importing Auxiliary Files on page 30](#)
- [Importing PDC Files \(Axcelerator family only\) on page 33](#)
- [Importing SDC Files on page 35](#)
- [Auditing Files on page 37](#)
- [Device Selection Wizard on page 39](#)
- [Compiling a Design on page 44](#)
- [Compile Options on page 45](#)
- [User Tools on page 47](#)
- [Layout on page 50](#)
- [Layout Failures on page 57](#)
- [Back-Annotation on page 58](#)
- [Generating Programming Files on page 60](#)
- [Changing Design Name and Family on page 65](#)
- [Exporting Files on page 67](#)
- [Generating Reports on page 72](#)
- [Setting Designer Preferences on page 83](#)
- [Starting other Applications from Designer on page 86](#)
- [Saving a Design on page 87](#)
- [License Details on page 88](#)
- [Ending the Designer Session on page 88](#)

Starting and Initiating a Design Session

Before you implement a design, you must start Designer and initiate a design session.

To start Designer:

From the Start menu, click Programs. From the Programs menu, select Designer and Click *Designer Series*.

Unix

Type the following command at the prompt:

<location of Actel software>/bin/designer

Designer opens and is ready for you to initiate your Designer Session, as shown in Figure 2-1.

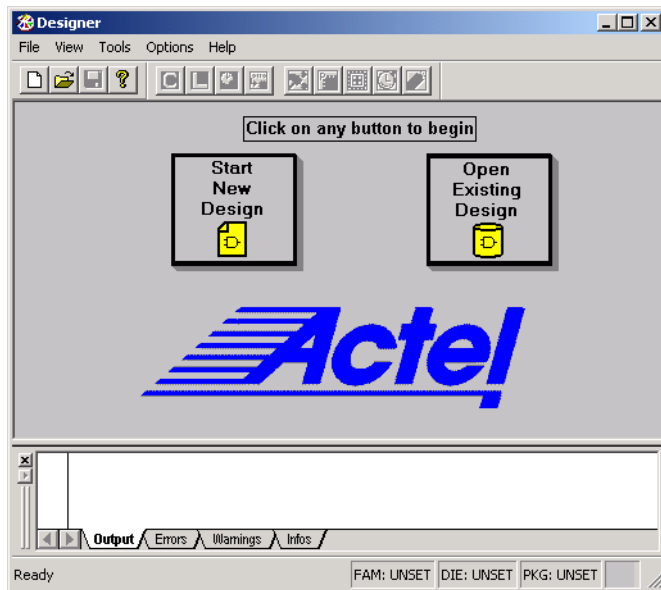


Figure 2-1. Starting Designer

After starting Designer, you are ready to begin your design session. Begin the design session by starting a new design or opening an existing design file.

Starting a New Design

1. Click the **Start New Design** button in the main window, or in the **File** menu, click **New**. This displays the Setup Design dialog box, as shown in Figure 2-2.

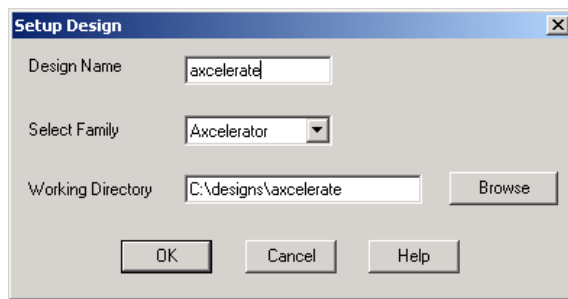


Figure 2-2. Setup Design Dialog Box

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 product Family from the drop down menu list.
- Specify a working directory. (Do not use spaces in the directory name.)

3. **Click OK.**

Designer's custom design flow window appears (as shown in Figure 1-4 on page 20). All Designer's tools and commands are activated.

Opening an Existing Design

Designer can open designs that have previously been saved, including designs from previous versions of Designer (refer to the next section, “Opening Designs Created in Previous Versions of Designer”).

To open an existing design:

1. **Click the *Open Existing Design* button or in the File menu, click *Open*.** This displays the Open dialog box (Figure 2-3).

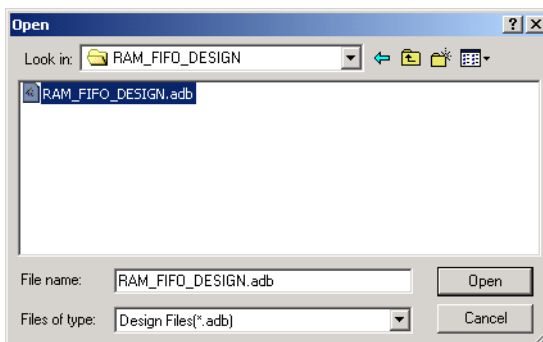


Figure 2-3. Open Design Dialog Box

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*.**

Note: 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.

Designer’s custom design flow window appears (as shown in Figure 1-4 on page 20) and all tools commands are activated.

Opening Designs Created in Previous Versions of Designer

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

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

All existing die, package, pin assignment, 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 launch any of the Designer features.

Importing Source Files

Design implementation begins with importing source files. Source files include your netlist and constraint files, such as the files in Table 2-1:

Table 2-1. Source Files

File Type	Extension
EDIF	*.ed*
Verilog	*.v
VHDL	*.vhd
Actel ADL Netlist	*.adl
Criticality	*.crt
Flash 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*.** This displays the Import Source Files dialog box, as shown in Figure 2-4.

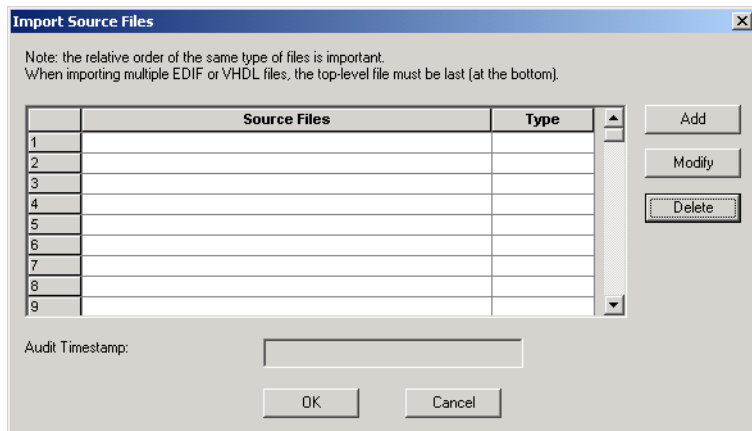


Figure 2-4. Import Source Files Dialog Box

2. **Click the *Add* button.** The Add Source Files dialog appears, as shown in Figure 2-5.

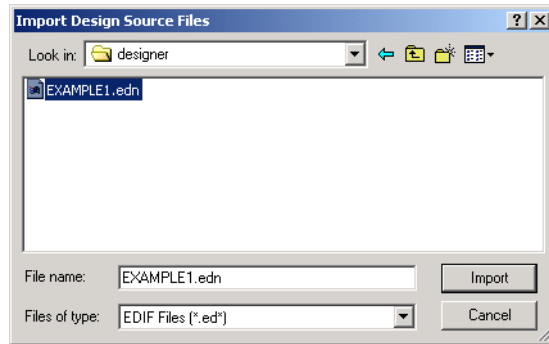


Figure 2-5. Add Source File Dialog Box

3. **Select your EDIF netlist and click *Import*.** The File is added to the Import Source Files dialog box, as shown in Figure 2-6.

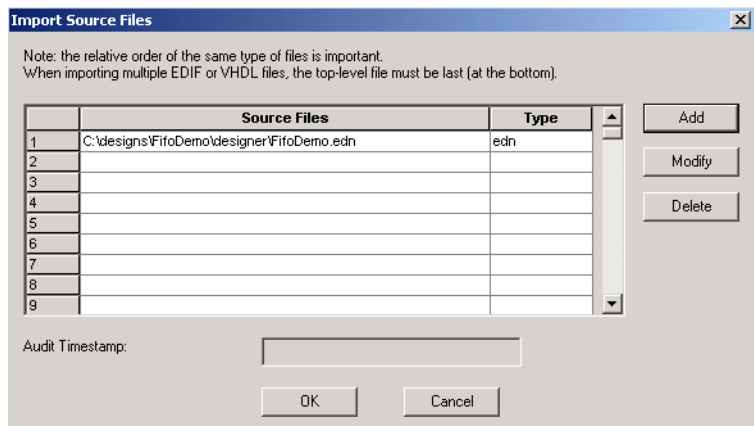


Figure 2-6. Import Source Files Dialog Box with EDIF File Added

4. **Add any more source files to the list.** All files added to the Import Source Files dialog box are imported at the same time.
If you need to modify a selection, select the file row and click *Modify*.

If you need to delete a file, select the file row and click *Delete*.

- 5. **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 Designer’s Log Window.

Note: Do not use spaces in your file or path names. Rename the file or path, removing the spaces, and re-import.

Importing Auxiliary Files

Auxiliary Files include are listed in Table 2-2:

Table 2-2. Auxiliary Files

File Type	Extension
Criticality	*.crt
PIN	*.pin
SDC	*.sdc
Physical Design Constraint	*.pdc
Value Change Dump	*.vcd
Switching Activity Interchange Format	*.saif
Design Constraint File	*.dcf

Note: .vcd and .saif are used by SmartPower for power analysis. Refer to the *SmartPower User’s Guide* for more details about performing power analysis.

Note: Criticality (.crt) is a legacy file format. It is supported for the following families: ACT1, ACT2, ACT3, 40MX, and 42MX.

To import an auxiliary file:

1. **From the File menu, click *Import Auxiliary Files*.** The Import Auxiliary Files dialog appears, as shown in Figure 2-7.

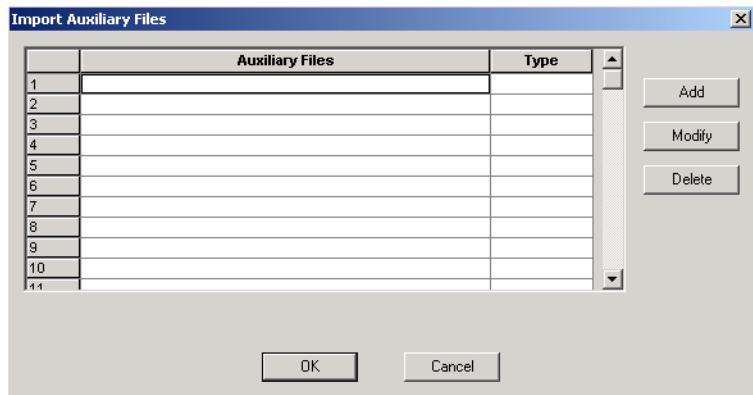


Figure 2-7. Import Auxiliary Files Dialog Box

2. **Click *the Add button*.** The Add Auxiliary Files dialog box appears, as shown in Figure 2-8.

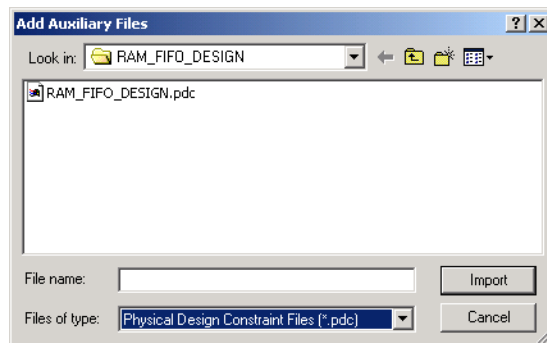


Figure 2-8. Add Auxiliary Files

Filter for files by using the Files of Type drop-down list box.

3. **Select your file and click *Import*. The file is added to the Import Auxiliary Files dialog box, as shown in Figure 2-9.**

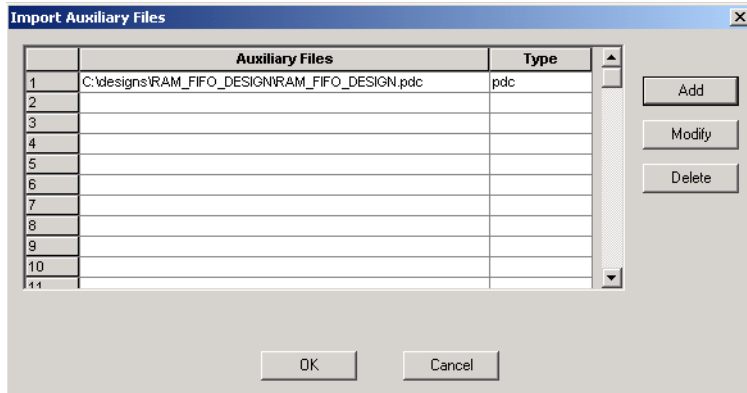


Figure 2-9. File Added to the Import Auxiliary Files Dialog

4. **Continue to add more auxiliary files to the list.**
 - To modify a selection, select the file row and click *Modify*.
 - To delete a file, select the file row and click *Delete*.
5. **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.
6. **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: 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 ChipEdit and PinEdit to set I/O standards and attributes, or alternatively you can export and import this information in a PDC file.

Physical Design Constraint (PDC) files are TCL script files. For information on TCL, see [“Scripting” on page 89](#). PDC commands, syntax, and examples are listed on [page 100](#).

PDC files are only supported for the Axcelerator family of devices. The PDC file replaces the PIN file.

To import a PDC file:

1. **From the File menu, click *Import Auxiliary Files*.** The Import Auxiliary Files dialog appears, as shown in Figure 2-10.

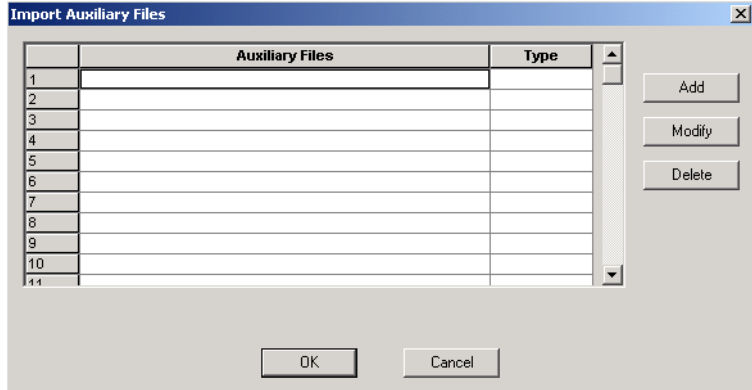


Figure 2-10. Importing a PDC File Dialog Box

2. **Click *the Add button*.** The Add Auxiliary Files dialog box appears, as shown in Figure 2-11.

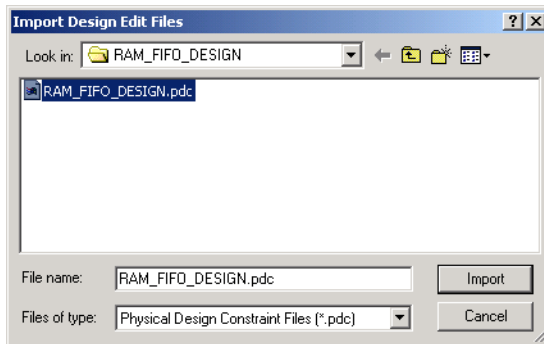


Figure 2-11. Selecting the PDC Files

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, as shown in **Figure 2-12**.

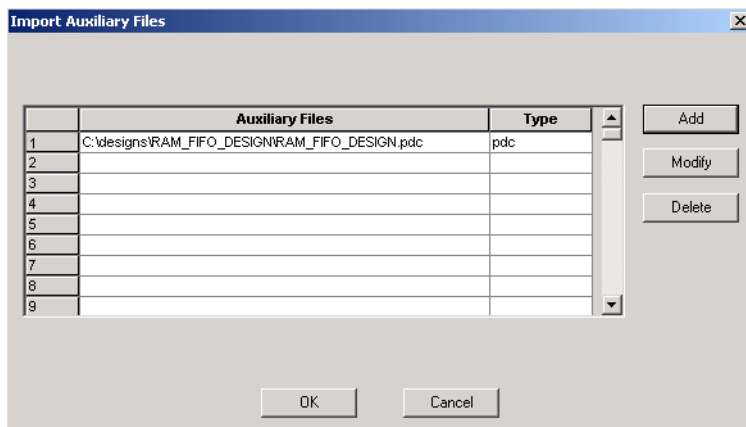


Figure 2-12. PDC File Added to the Import Auxiliary Files Dialog

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.

Importing SDC Files

Synopsys Design Constraints (SDC) files can be imported into Designer, to be read by Timer. SDC is a widely used format that allows designers to utilize the same sets of constraints to drive synthesis, timing analysis, and place-and-route.

SDC is a Tcl based format constraint 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 the user can generate the SDC file manually.

For more information on supported SDC commands and limitations, see “[SDC Commands](#)” on page 104.

Importing SDC Files

To import an SDC file:

1. **From the File menu, click *Import Auxiliary Files*.** The Import Auxiliary Files dialog box is displayed, as shown in Figure 2-13

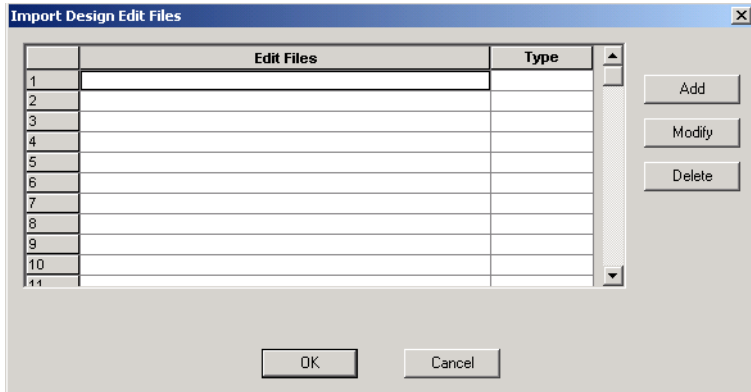


Figure 2-13. Import Auxiliary Files Dialog Box

2. **Click *Add*.** The Add Auxiliary Files dialog box appears, as shown in Figure 2-14 .

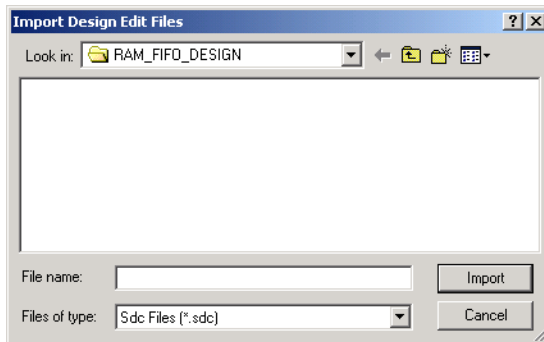


Figure 2-14. Adding an SDC File

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* in the Import Auxiliary Files dialog box.** The SDC file is imported into your design. 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.

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, as shown in Figure 2-15.

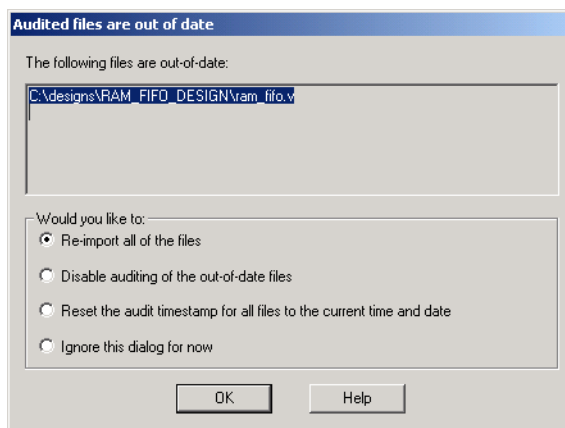


Figure 2-15. Audited File is Out of Date Dialog Box

When notified, select the appropriate action and click *OK*. To disable auditing, follow the steps below.

To change your audit settings:

1. **From the File menu, click *Audit Settings*.** The Audit Settings dialog box appears, as shown in Figure 2-16.

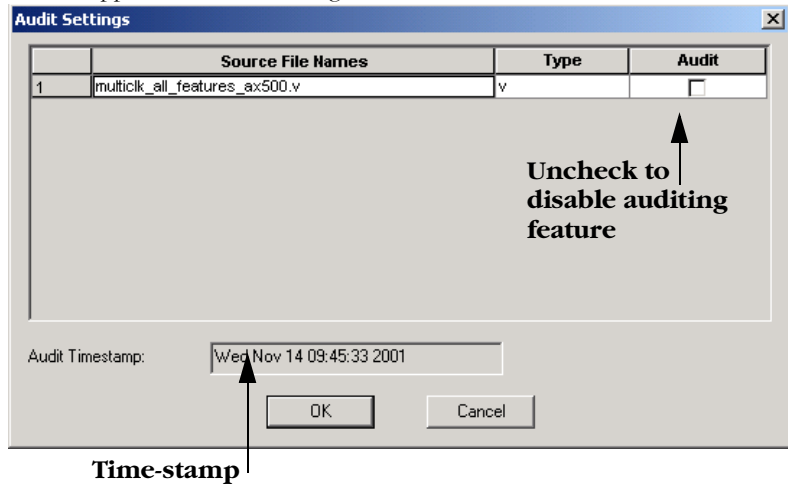


Figure 2-16. Update Audit Source Files Dialog Box

Audit Timestamp reflects the last time/date that the import source or audit update was successfully done.

2. **Disable auditing by un-checking the audit check box next to the file.**

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. Starting compile without completing the device selection automatically starts the Device Selection Wizard.) To change this information for existing designs, refer to “Changing Design Name and Family” on page 65 and “Changing Design Information” on page 66.

To select device, package, and other operating conditions:

1. **In the Tools menu, click *Device Selection*.** The Device Selection Wizard starts, as shown in Figure 2-17.

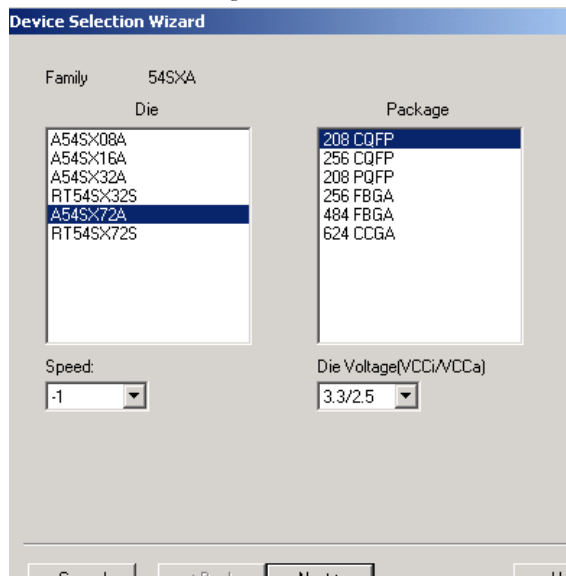


Figure 2-17. Device Selection Dialog Box

2. **Select die and package.** Select a die from the Die list. Available packages are listed for each die. Select a package.
3. **Specify speed and die voltage.** Select from the available settings in the Speed Grade and Die Voltage drop-down menus. 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.

4. **Click *Next*.** The Device Selection Wizard prompts you to set Variations, as shown in Figure 2-18.

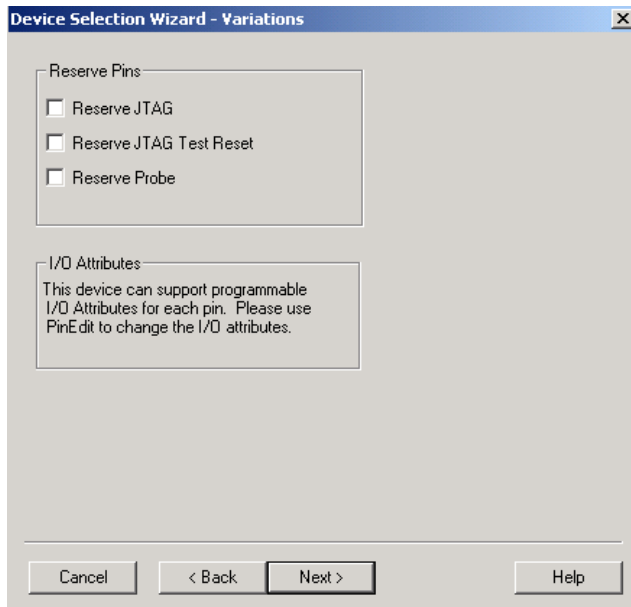


Figure 2-18. Device Selection Wizard, Device Variations (Screen Varies Depending Upon Device)

5. Set device variations.

Note: Reserve Pins are not selectable for the Axcelerator, ProASIC, and ProASIC PLUS families.

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. (Probe pins are family dependent.)

The I/O Attributes section notifies you if your device supports the programming of I/O attributes on a per-pin basis.

For the Accelerator family, the I/O Attribute section allows you to set the default I/O standard for the I/O banks, as shown in Figure 2-19.

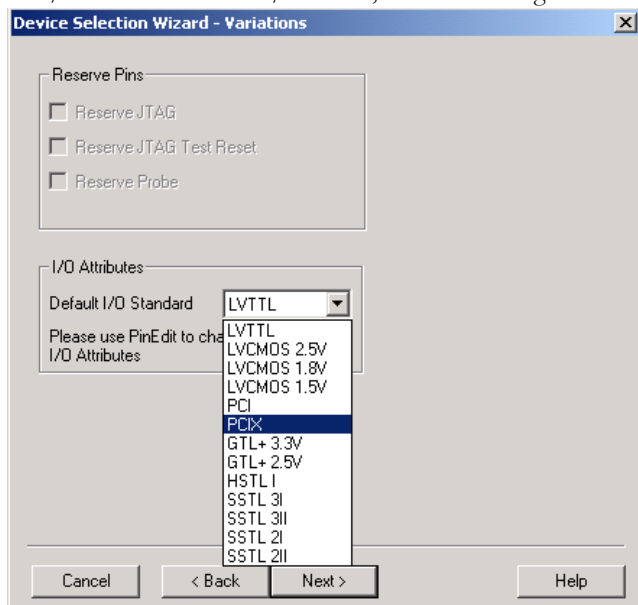


Figure 2-19. Device Selection Wizard - Variations for the Accelerator Family

Use PinEdit to assign different I/O standards to different I/O banks, if necessary. (Refer to the *PinEdit User's Guide* for more information on assigning I/O attributes.)

6. **Click *Next*.** The Device Selection Wizard prompts you to set the Operating Conditions.

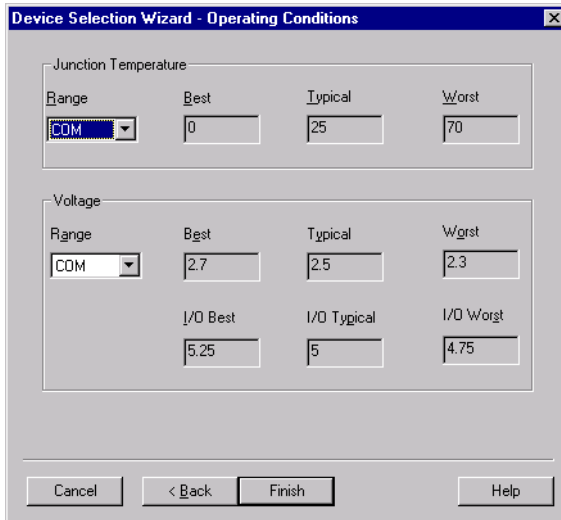


Figure 2-20. Device Selection Wizard, Operating Conditions

7. **Set Operating Conditions and Click *Finish*.** Use the Operating Conditions dialog box (see Figure 2-20) to define the voltage and temperature ranges a device encounters in a working system. Supported ranges include standard industry temperature and voltage ranges, including commercial (COM), industrial (IND), and military (MIL). This displays supported ranges. Select Custom in the pull-down menu to specify a custom range. The operating condition range entered in the Operating Conditions dialog box is used by Timer, the timing report, and the back-annotation timing function. These tools enable you to analyze worst, typical, and best case timing. The operating conditions are summarized in Table 2-3.

Table 2-3. Operating Conditions

Timing	Process	Temperature	Voltage
Best Case	Best	Best	Best
Typical Case	Typical	Typical	Typical
Worst Case	Worst	Worst	Worst

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. You can calculate junction temperature from values in the Actel *Data Sheet*, available at <http://www.actel.com/techdocs/ds/index.html>. Performance decreases approximately 2.5% for every 10 degrees C that the temperature values increase. For Axcelerator, ProASIC, and the ProASIC^{PLUS} families please use SmartPower for more accurate power consumption estimation. Refer to the *SmartPower User's Guide* for more information about power consumption.

Temperature Range

Select a supported temperature range from the pull-down menu (COM, IND, MIL or Custom). If you select Custom, edit the Best, Typical, and Worst fields. Modify the range to the desired value (real) such that Best \leq Typical \leq Worst.

Voltage Range

Select a voltage range from the pull-down menu (COM, IND, MIL or Custom). If you select Custom, edit the Best, Typical, and Worst fields. Modify the range to the desired value (real) such that Best \geq Typical \geq Worst. The top row indicates core (array) voltage, or both core and array voltages if they are the same. The lower row shows the I/O voltage for mixed voltage devices, and is ignored for non-mixed voltage devices.

Compiling a Design

After you import your netlist file(s) 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. See [“Device Selection Wizard” on page 39](#) for information on the Device Selection Wizard.

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 (PinEdit, ChipEdit, ProASIC Layout Viewer, Timer, SmartPower, or Netlist Viewer) 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, as shown in Figure 2-21 and as shown in Figure 2-22. Options available from this dialog box are dependent upon your family.
2. **Select your options and click *OK*.** Options are explained below.

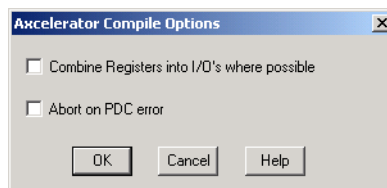


Figure 2-21. Compile Options Dialog Box, Accelerator Family

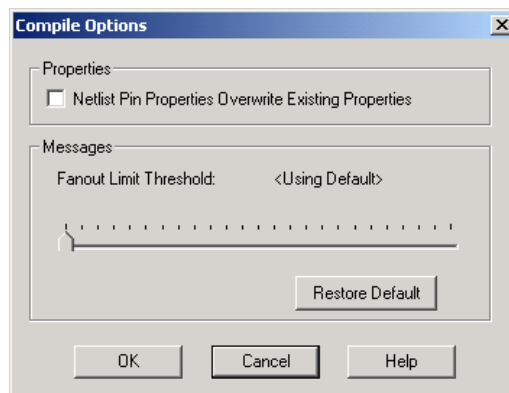


Figure 2-22. Compile Options Dialog Box, SX Family

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 PinEdit 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 PinEdit 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 PinEdit assignment is ignored.

If you edit pin assignments in PinEdit, this option is automatically set to "off."

Combine Registers into I/Os

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 only. 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

Use the control slider in the Messages area to control the warning level for fanout. 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.

Note: This option is available for non Axcelerator, ProASIC and ProASIC PLUS.

User Tools

After importing and compiling your design, you can, if necessary, optimize and customize your design with the User Tools before running Layout. The User Tools include PinEdit, ChipEdit, ChipView, Netlist Viewer, SmartPower, Timer, and Back-Annotate. Below is a brief summary of the User Tools. Specific details on using these tools can be found in their respective User's Guides.

PinEdit

Use PinEdit to customize I/O assignments and attributes.

There are two methods for I/O signal placement. You can let Designer automatically assign I/O locations during Layout, or you can manually assign I/O locations prior to Layout.

For non-Axcelerator families Actel recommends that you let Designer automatically assign I/O locations during Layout. You can then use PinEdit to optimize your design, if needed. Layout is designed to place the I/Os for optimum routability and performance. Refer to [“Layout” on page 50](#) for information about automatically assigning I/O locations during place-and-route.

When targeting the Axcelerator family and individual I/O bank configuration is needed, *you must use PinEdit to assign I/O standards to each bank before running Layout.*

Manually assign I/O locations in your design schematic, in imported files, or by using PinEdit. Imported files can include PIN files (non-Axcelerator families), PDC files (Axcelerator family only), or GCF files (ProASIC and ProASIC^{PLUS} families only).

Refer to documentation included with your CAE tools for information about assigning I/O signal placement in a schematic or in a pin file. Refer to the *PinEdit's User's Guide* for more information on using PinEdit.

ChipEdit

Use ChipEdit to view and edit the placement of both I/O and logic macros. Refer to the *ChipEdit's User's Guide* for more information on using ChipEdit.

Note: For the Axcelerator family, you must use ChipEdit before running Layout to place the I/O FIFO Block Controllers. ChipEdit does not support the ProASIC and ProASIC^{PLUS} families.

ChipView (ProASIC and ProASIC^{PLUS} families only)

For the ProASIC and ProASIC^{PLUS} families, ChipView displays the results of place-and-route. These results provide information to guide later place-and-route operations, if necessary. The window creates no new data. It displays the design layout and is used for identifying problems and providing insights to solve them. Refer to the *ChipEdit's User's Guide* for more information on using the ChipViewer.

Timer

Timer performs static timing analysis on your design. Use Timer to analyze timing performance and set timing constraints. If you want to run Timing-Driven Layout, you must use Timer to set and commit timing constraints.

Netlist Viewer

The Netlist Viewer displays a graphical representation of the netlist. Use it in conjunction with ChipEdit and Timer to locate objects and to trace paths. Refer to the *Netlist Viewer User's Guide* for more information.

SmartPower

SmartPower calculates the power consumption of the currently loaded design. Refer to the *SmartPower User's Guide* for more information.

Back-Annotate

The backannotation function is used to extract timing delays from your post layout data. These extracted delays are put into a file to be used by your CAE packages timing simulator. Refer to [“Back-Annotation” on page 58](#) of this User's Guide for more information.

Place-and-Route Variables (Non-ProASIC and ProASIC PLUS Families)

Use the Set Variable dialog box to set a variable or use Actel scripting to set variables. (For information in scripting commands, refer to “Scripting” on page 89.) Variables must be set before running layout.

To set a variable:

1. **From the Options menu, click *Set Variable*.** The Set Variable dialog box appears as shown in Figure 2-23.

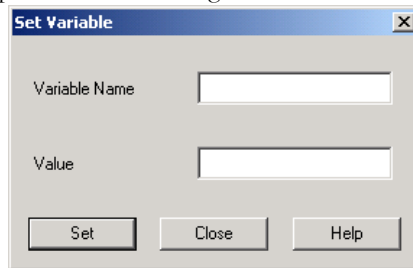


Figure 2-23. Set Variable Dialog Box

2. **Enter a Variable Name and Value.** Variables and their values include the following:

PLACESEEDRANDOM

Place-and-route uses several different inputs to determine where to start with the algorithms. Some of the inputs include: pin placement, device and package, previously placed macros, timing constraints, etc. In addition, there is another user-definable input called the SEED. This variable enables you to get a different result without changing any of the other place-and-route inputs. This can be valuable for designs that marginally fail to place-and-route. Set this variable to any value between 1 and $(2^{31}) - 1$ before running Layout. A script to automatically run Layout with different seeds can be found in “Extended Layout” on page 118.

PLACEACCEPTHINTS (Non-Axcelerator Families)

This variable enables you to use the contents of a .loc file as a starting point for layout. The .loc file contains all the placement information for a design. Using this variable is similar to using incremental ON, and should only be used as a last resort for design conversion situations. Make sure the .loc file has the same name as your design and place it in the directory of the design you are working on. Then set this variable to '1' and run Layout.

Layout

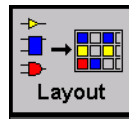
After you compile, the design is ready for layout. Layout takes the netlist information and any constraints and maps this information into the selected Actel device. Layout assigns physical locations to unassigned I/O and logic modules (placement), routing tracks to nets (routing), and calculates detailed delays for all paths (extract).

Designer supports two modes of layout, Standard and Timing-Driven. The physical result of each approach is similar, but the tools and algorithms are quite different. In either mode, the incremental placement option allows you to save the performance of a successfully placed and routed design, even if you change the netlist.

To layout your design.

1. There are three ways to initiate the Layout command:

- In the Tools menu, click *Layout*.
- Click the Layout button in the Design Flow.



- Click the Layout icon in the toolbar.



This displays the Layout Options dialog box, as shown in Figure 2-24.

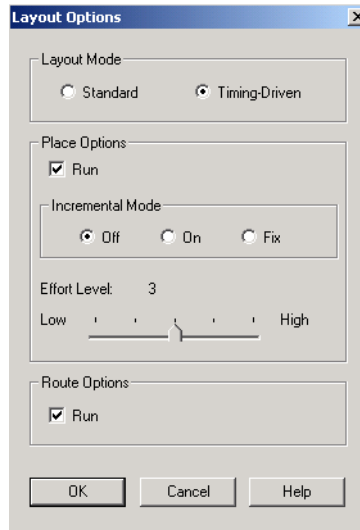


Figure 2-24. Axcelerator Layout Options Dialog Box

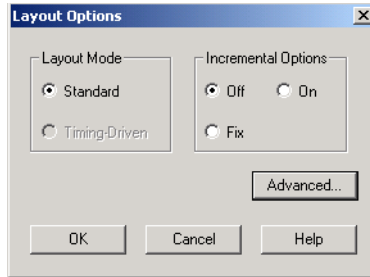


Figure 2-25. Other Anti-fuse Families Layout Options Dialog Box

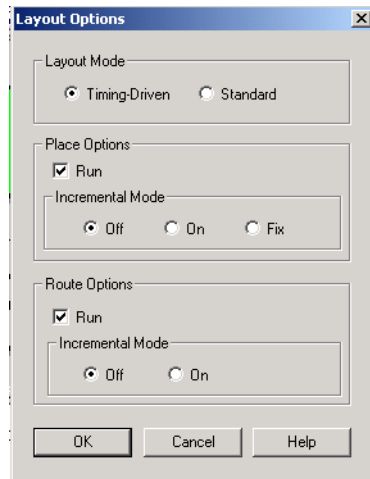


Figure 2-26. ProASIC and ProASIC ^{PLUS} Options Dialog Box

2. **Set your Layout options.** Options are family dependent. See [“Layout Options” on page 53](#) for a complete description of these options.

3. **Set your Advanced Layout options (Optional).** Click the *Advanced* button in the Layout dialog box. This displays the Advanced Layout Options dialog box, as shown in Figure 2-27.

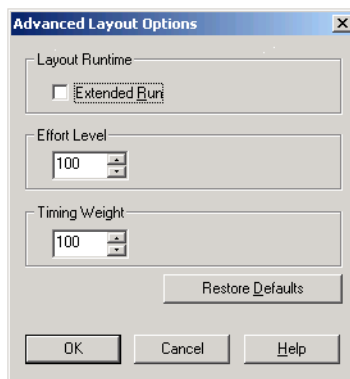


Figure 2-27. *Advanced Layout Options (SX, SX-A, and eX)*

Use the Advanced Layout Options dialog box to select extended run, specify an effort level, and set a timing weight. See [“Layout Options” on page 53](#) for a complete description of these options.

4. **Click *OK*.** Layout runs. Status messages appear in the Log window.

Layout Options

The options available in the Layout and Advanced Layout dialog boxes are family dependent. Below is a description of the various options.

Layout Mode

Standard

Standard layout maximizes the average performance for all paths. Standard layout treats each part of a design equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results.

Standard layout does not consider delay constraints that have been set for a design during place-and-route. 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.

Timing-Driven

The primary goal of Timing-Driven layout is to meet delay constraints set in Timer, SDC files (Axcelerator family only), DCF files (non-Axcelerator families), and GCF files (ProASIC and ProASIC ^{PLUS}) For information on GCF files, refer to “ProASIC Timing Constraints” on page 121.

Timing-Driven layout’s secondary goal is to produce high performance for the rest of the design. Delay constraint driven design is more precise and typically results in higher performance.

Note: Timing Driven Layout is available after you have entered timing constraints.

Place Options

Run

The Run checkbox selects whether the placer runs during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run is checked by default. If your design has already been placed, this box is not checked.

Incremental Placement Mode

- Off: No previous placement data is used.
- On: Previous placement data is used as the initial placement for the next place run.
- Fix: Previous placement data is used and is fixed for the next place run.

In either Standard or Timing-Driven mode, the Incremental Mode option allows you 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 Layout. Do not use incremental placement if the previous Layout failed to meet performance goals.

The FIX setting treats all unchanged macros as fixed placements. 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 the *ChipEdit User's Guide* for details on fixing macros.

For ProASIC and ProASIC ^{PLUS} designs, Designer 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_placement.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. Other wise, it is overwritten by any subsequent placement if it is left in its original location.

Note: For information on .gcf files, refer to "ProASIC Timing Constraints" on page 121.

Effort Level (Axcelerator Family)

Use the Effort Level slider to increase or decrease the amount of time you want Layout to run. A higher effort level runs Layout for a longer period of time and generally improves the quality of results. The default level is 3.

Route Options

Run

Selecting the Run checkbox runs the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run is checked.

Run is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this box is checked.

Incremental Mode

- Off: No previous routing data is used.
- On: Previous routing data is used as the starting point for the next router run.

Incremental routing allows you 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.

Advanced Layout Options

Extended Run

Extended run attempts to improve layout quality by using a greater number of iterations during optimization. An extended run layout can take up to 5 times as long as a normal layout. For scripting information, refer to [“Extended Layout” on page 118](#).

Note: This advanced option is available for all Antifuse Families.

Effort Level (SX, SX-A, and eX Families)

This variable specifies the duration of the timing-driven phase of optimization during layout. Its value specifies the duration of this phase as a percentage of the default duration.

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.

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

Timing Weight

Setting this option to values within a recommended range of 10-150 changes the weight of the timing objective function, thus biasing TDPR in favor of either routability or performance.

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.

Note: This advanced option is only available for the SX, SX-A, and eX families

Layout Failures

If Layout fails at any stage, Designer provides information that can help you determine and correct the problem. This section describes some failures and methods to fix the failures.

Failures During Timing Driven Layout

Layout can fail during initialization if the assigned constraints are impossible (i.e. no routing path on the device can meet the assigned constraint). You must

change the circuit or relax the constraints to proceed with Timing-Driven Layout.

- Change the Speed Grade to increase minimum delay.
- Modify the design to reduce the number of logic levels in these paths.
- Relax over-restrictive delay constraints. If the constraints from Timer or a DCF file are unnecessarily tight, change them to more realistic values that still satisfy your timing requirements.
- Fixed pin placements may not allow Layout to succeed. Unplace or unfix I/O pins.

Back-Annotation

The backannotation 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.

To back-annotate your design:

1. **From the Tools menu, click *Back-Annotate*, or click the *Back-Annotate* button in the Design Flow window.** This displays the Back-Annotate dialog box, as shown in Figure 2-28.

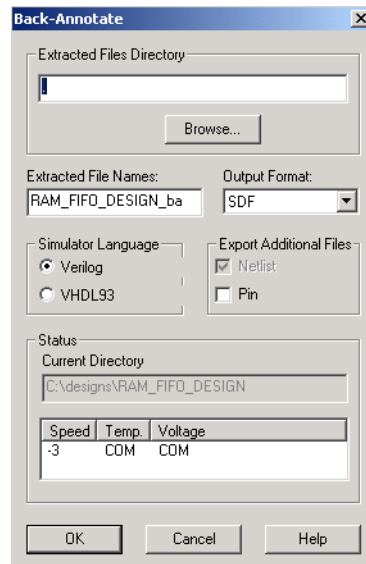


Figure 2-28. Back-Annotate Dialog Box

2. **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.
3. **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 is assigned based on your selection of which file formats to export. The default value of this field is <design>_ba.
4. **Select Output format.** Select an output format the Output Format drop-down menu.

5. **Select a Simulator Language.** Select either Verilog or VHDL93. Simulator Language becomes available when you select SDF as an output format.
6. **Export Additional Files.** Check Netlist or Pin to export these files at the same time.
7. **Click OK.** 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.

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.

Table 2-4. Programming Files

Programmer	Antifuse Programming File	Flash Programming File
Flash Pro	N/A	STAPL
Silicon Sculptor I	Fuse (Non-Axcelerator families)	Dos and Windows: Bitstream
Silicon Sculptor II	Fuse (All families)	Dos Version: Bitstream Windows Version: Bitstream or STAPL

Fuse

Fuse allows you to generate a programming file for your design to program an Actel device.

Silicon Signature (Antifuse Devices only)

You can specify a unique silicon signature to program into the device when you generate a programming file. This signature is stored in the design database, the programming file, and programmed into the device permanently during programming. With Designer tools, you can use the silicon signature to identify and track Actel designs and devices.

To generate a programming file:

1. **In the Tools menu, click *Fuse* or click the Fuse button in the Design Flow window.** This displays the Generate Fuse file dialog box (Figure 2-29).

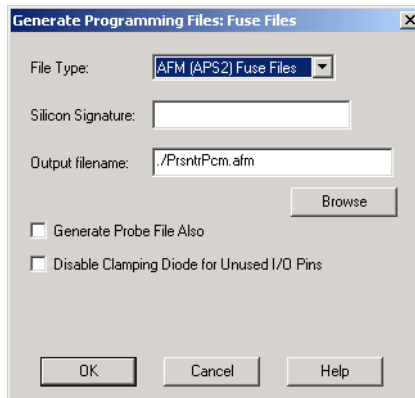


Figure 2-29. Generate Fuse File Dialog Box

2. **Specify File Type.** Select the appropriate file type in the File Type pull-down menu. Select “AFM-APS2” if you are using a Silicon Sculptor programmer.
3. **Enter 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.”

4. **Specify File name and directory in the Output filename box.** 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.

Note: 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.
5. **Generate Probe File Also.** This option automatically generates a .prb file for use with Silicon Explorer
6. **Disable clamping diode for unused I/O pins. (SX-A and eX families).** Check box to disable clamping diode.
7. **Save the file.** Click *OK* when finished to save the file.

Bitstream and STAPL Files



Bitstream allows you to generate a bitstream or STAPL file for ProASIC and ProASIC^{PLUS} devices. 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 options:

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. Table 2-5 and Table 2-6 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.

Table 2-5. Key Size of Pro.ASIC Devices

Device	Key Size (bits)	Key Size (Hex)
A500K050	51 Bits	13
A500K130	51 Bits	13
A500K180	51 Bits	13
A500K270	51 Bits	13

Table 2-6. Key Size of Pro.ASIC^{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

The maximum security key for the device is shown in the dialog box.

To generate a bitstream or STAPL file:

1. In the Tools menu, click **Bitstream** or click the **Bitstream** button in the Design Flow window. This displays the Bitstream dialog box (as shown in Figure 2-30).

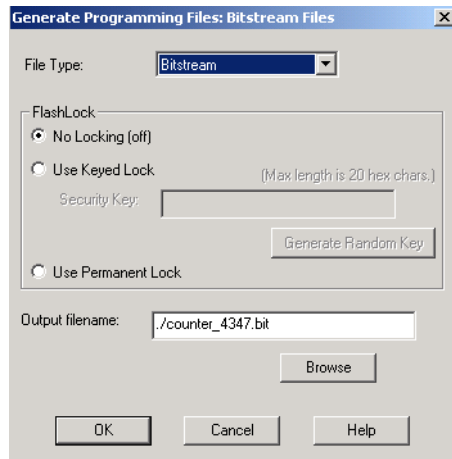


Figure 2-30. Bitstream Generation Dialog Box

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 (see “Keyed Lock” on page 62).
 - 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.

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.

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, as shown in Figure 2-31..

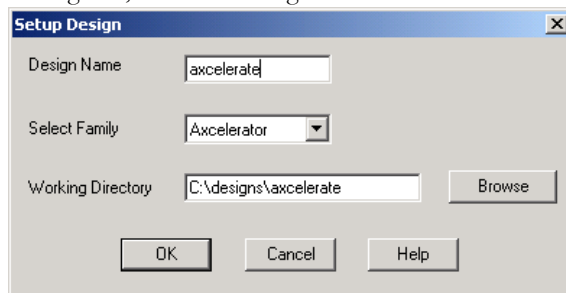


Figure 2-31. Setup Design Dialog Box

2. **Specify the design name and family.** Click *OK*. Refer to the Actel *FPGA Data Book* for Actel Family specifications.

Changing Design 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.** (You must select die and package to continue.) Click *Next*.
3. **Select Device Variations.** Click *Next*.
4. **Select Operating Conditions.** 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 (see Figure 2-17 on page 39) to specify or change the device and package type and the speed grade based on your design needs..

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-Accelerator 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.

Exporting Files

Designer lets you export auxiliary files (.afl, .bsd, .cob, .crt, .dcf, .gcf, .loc, .pdc, .pin, .prb, .saif, seg, .vdc), fuse files (.afm, .dio, .fus), bitstream files (.bit), log files (.log), netlist files (.adl, .edn, structural VHDL, STAPL files (.stp), and structural Verilog), script files (.tcl), stamp files (*.mod and *data), and delay files (.stf and .sdf) from your design.

Note: Designer does not support VHDL 87 in export.
Exported file types are family dependent.

To export a file:

- 1. In the File menu, click an export option from the *Export* sub-menu. Select *Netlist*, *Auxiliary*, *Fuse*, *Bitstream*, *Timing*, *Script*, or *Log* files.** This displays a dialog box specific to the file type.
- 2. Specify file name and file type.**
- 3. Click *OK*.**

Exporting Bitstream Files

For important information on Bitstream and STAPL files, read “[Bitstream and STAPL Files](#)” on page 62.

To export a bitstream file:

1. **In the File menu, click the *Bitstream* export option from the Export sub-menu.**
2. **Specify file name and file type.** Designer supports two file types, bitstream (*.bit) for Silicon Sculptor, and STAPL (*.stp) files for programming systems that support the STAPL standard.
3. **Click *OK*.** This displays the Bitstream Export Options dialog box, as shown in Figure 2-32.

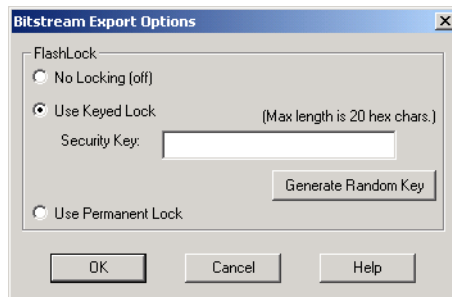


Figure 2-32. Bitstream Export Options Dialog Box

4. **Read and follow the instructions in “[Bitstream and STAPL Files](#)” on page 62.**

Exporting PDC Files

To export a Physical Design Constraint (PDC) file:

1. **From the File Menu, select *Export*, and click *Auxiliary Files. . .* from the *Export* sub-menu.** The Export Auxiliary Files dialog box appears, as shown in Figure 2-33

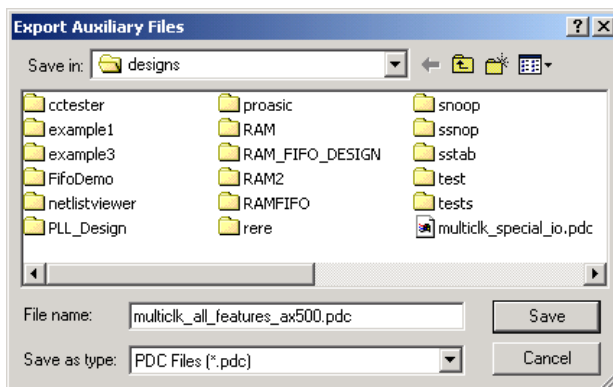


Figure 2-33. Exporting Your PDC File

- Save in: Navigate to the drive and folder where you want to save the file.
- File Name: Type your file name.
- Save As Type: Select PDC.

2. **Click *Save*.**

Exporting STAMP files

Designer supports Stamp model generation for the eX, Accelerator, SX-A, and RTSX-S families. Stamp is an industry-standard file format that contains timing data. Stamp models are read by board timing verification tools, such as Mentor Graphics's TAU and Viewlogic's BLAST.

The Stamp model for a design consists of two files:

- The *.mod file describes the Interface (IOs) and all timing arcs of the model.
- The *.data file gives a timing value for each timing arc defined in the model.

Designer supports the following subset of Stamp modeling language for best, typical, and worst operating conditions:

- Inputs to Outputs propagation delay arcs, for shortest and longest paths.
- Inputs setup and hold timing-check arcs.
- Clock period timing-check arcs.

To export a STAMP file:

1. **In the File Menu, click the *Timing* export option from the **Export sub-menu**.** This displays the Export Timing Files dialog box, as shown in Figure 2-34.

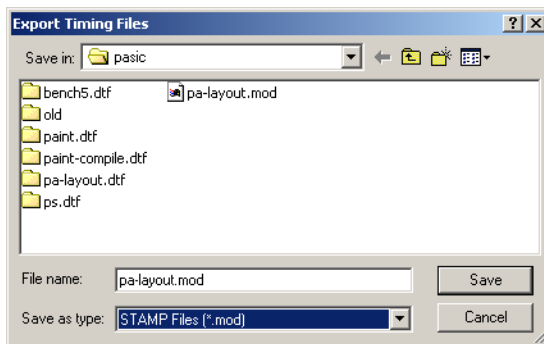


Figure 2-34. Export Timing Files Dialog Box

2. **Select *STAMP* from the Save As Type list box.**
3. **Enter a file name in the File Name text box, followed by *.mod*, and click *Save*.** The Preferences dialog box appears, as shown in Figure 2-35.

Note: The corresponding *.data file is generated with the same name.

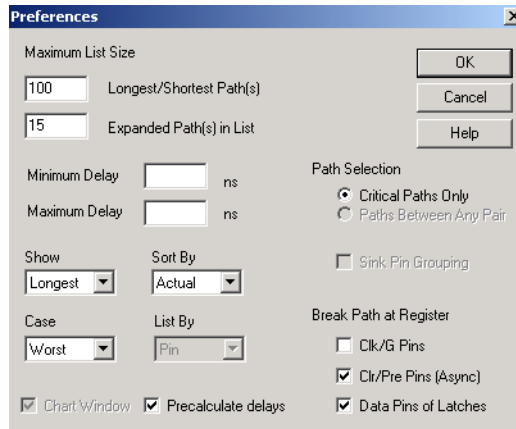


Figure 2-35. Timing Preferences Dialog Box

- 4. Set your preferences and click OK.** Timer exports the file to the area specified in the Export Timing Files dialog box. For information on setting preferences, please refer to the *Timer User's Guide*.

Note: Inputs to Outputs /Shortest/Longest delays cannot be generated in the same data file. You must output two separate Stamp files.

Longest Paths / Shortest Paths selection in the Preferences dialog box only affects Inputs to Outputs propagation delays and does not affect Period, Setup, and Hold timing-check delays.

Operating conditions potentially affect all timing values of the Stamp data file. It is necessary to export 3 separate Stamp files if best/typical/worst models are needed.

Generating Reports

Designer's reports provide you with frequently used information in a convenient format.

To generate a report:

1. **In the Tools menu, click *Reports*.** The Report Types dialog box appears.

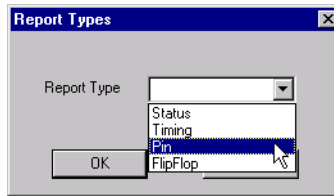


Figure 2-36. Report Type Dialog Box

2. **Select a report type from the drop-down menu and click *OK*.**

Status Report

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 Report

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.

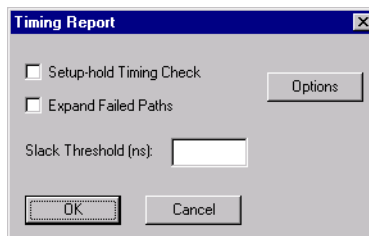


Figure 2-37. Timing Report Dialog Box

- 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.

- Options.** Clicking *Options* brings up the Timing Preferences dialog box, where you can set additional display and report options.

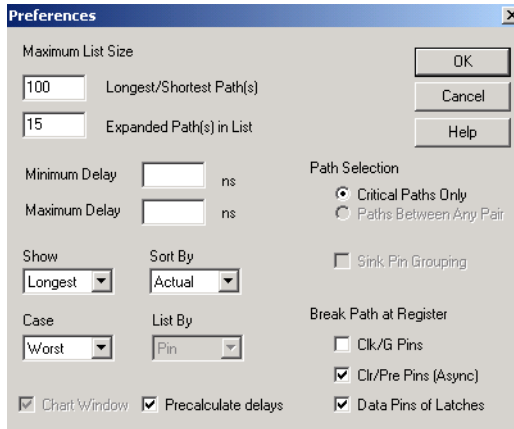


Figure 2-38. Timing Preferences Dialog Box

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 automatically displays the slack - even if you don't ask for it - but the data is always 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 is 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, as shown in Figure 2-39. The format and content of the report is determined by the family.

```
Timer Version 01.01.01
Actel Corporation - Designer Series (tm) Development System Re
Copyright (c) 1989-2001
Date: Wed Nov 07 15:55:42 2001

Design: snoop_arb
Family: 545XA
Die: A545X08A
Package: 208 PQFP
Temperature: COM
Voltage: COM
Speed Grade: -3
Design State: Post-Layout
Timing: worst Case
Path Tracing: Longest Paths
Break at CLK/G pins: True
Break at Preset/Clr pins: True
Break at Data pins of Latches: True

Section Clock Frequency
      Actual      Required      ClockName
      146.65Mhz   -1.00ns      CLK
End Section

Section $Inputs() to $Outputs()
      Delay(ns)  Slack(ns)  Pins
      7.42      N/A        From: frame_1
                        To: reset_out_1<0>
      6.38      N/A        From: frame_1
                        To: reset_out_1<1>
      6.26      N/A        From: indy_1
                        To: reset_out_1<0>
      5.99      N/A        From: frame_1
                        To: reset_out_1<2>
      5.99      N/A        From: indy_1
                        To: reset_out_1<2>
      5.55      N/A        From: indy_1
```

Figure 2-39. Timing Report (Accelerator, SX-A, eX, ProASIC, and ProASIC Plus families)

Timing Violations

To generate a timing violations report:

1. From the Tools menu, click Reports.
2. In the Reports dialog box, select Timing Violations and click OK.
3. The Timing Violations report appears in a separate window.

Pin Report

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.

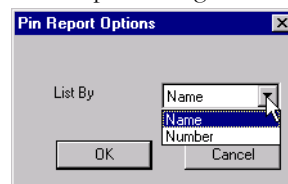


Figure 2-40. Pin Report Dialog Box

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, as shown in Figure 2-41.

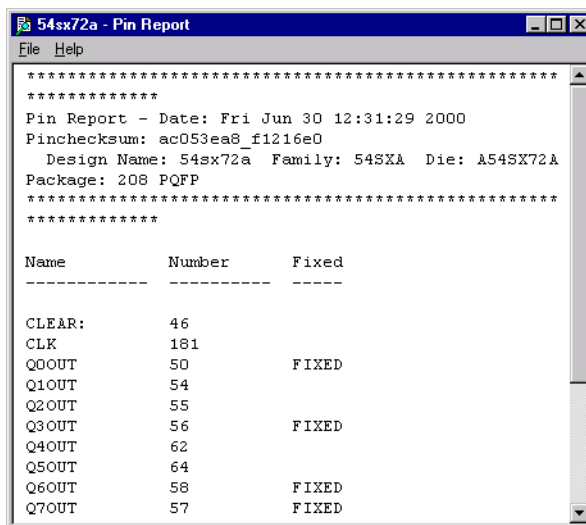


Figure 2-41. Pin Report

Flip-Flop Report

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, as shown in Figure 2-42.

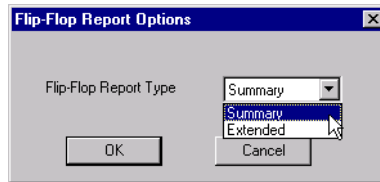


Figure 2-42. Flip-Flop Report Dialog Box

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, as shown in Figure 2-43.

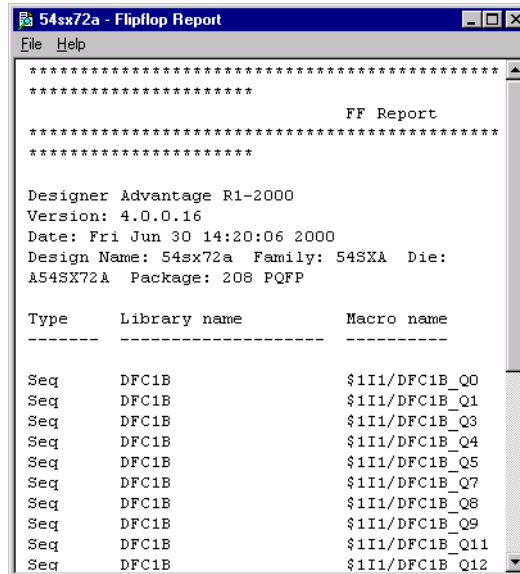


Figure 2-43. Flip-Flop Extended Report

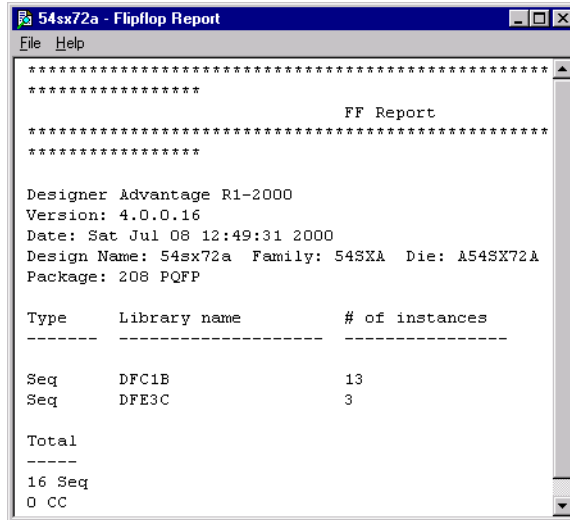


Figure 2-44. Flip-Flop Summary Report

Power Report

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, as shown in Figure 2-45.

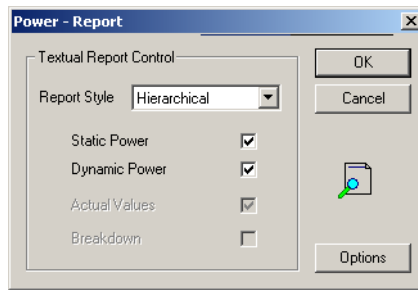


Figure 2-45. Power Report Dialog Box

You have the following options:

- **Static Power:** Returns static power information
- **Dynamic Power:** Returns dynamic power information
- **Report Style:** Specifies report style

3. For additional Power Report Options, click the *Options* to open the Power Preferences dialog box, as shown in Figure 2-46.

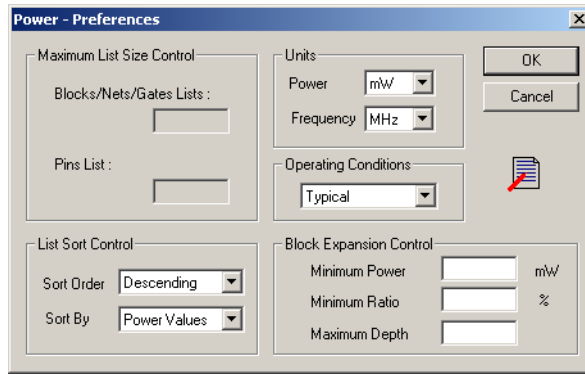


Figure 2-46. Power Preferences Dialog Box

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.

Setting Designer Preferences

Setting the Start-up Directory

You can set the start-up directory for Designer. Whenever you execute a command or function such as *Open* or *Save*, Designer uses the directory you specify as the start-up directory.

To specify directory preferences:

1. **From the File menu, click *Preferences*.** The Program Preferences dialog box appears, as shown in Figure 2-47.

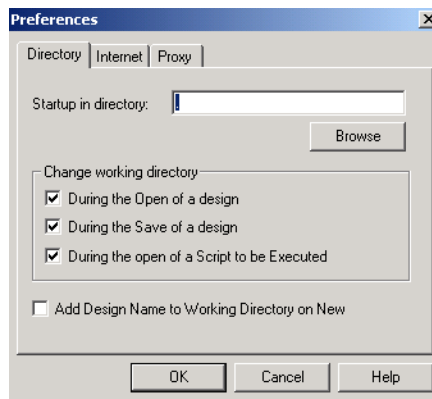


Figure 2-47. Directory Preferences Dialog Box

2. **Start-up in directory:** Click *Browse* to change your default working directory.
3. **Change your working directory.** Enable or disable the ability to automatically change directories during the opening of a design, the saving of a design, or the opening of a script.
4. **To enable a design name folder to be automatically created in the working directory whenever you create a new design, check Add Design Name to Working Directory on New.**
5. **Click OK.**

Internet Features

Use the internet tab in the Preferences dialog box to set your automatic version checking preferences (Figure 2-48). You can enable Designer to automatically check if there is a new version at start-up, prompt you before checking, or disable version checking entirely.

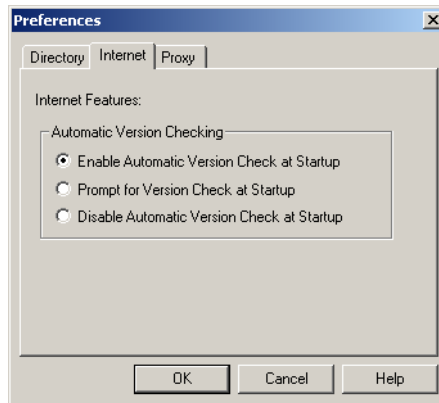


Figure 2-48. Internet Tab (in Preferences Dialog Box)

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. From the File menu, click Preferences and then the Proxy tab.

File Association (PC only)

Several programs, including Designer, create files with the .adb extension. Use the File Association 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*. The Preferences dialog box appears.**

2. Click **File Association**, as shown in Figure 2-49.

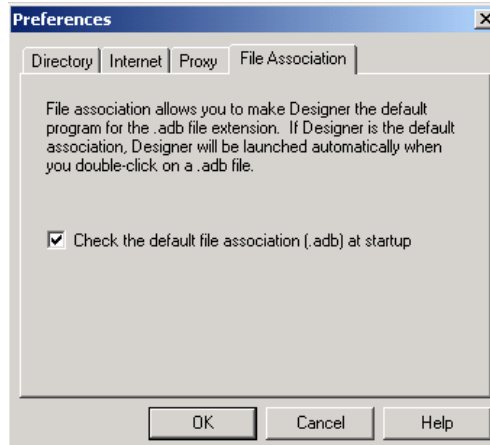


Figure 2-49. File Association Tab

3. 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.
4. Click **OK**.

PDF Reader (Unix Only)

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

Web Browser (Unix Only)

The Web Browser tab is used to change the default web browser. Enter the new browser's name with the full path, or click *Browse*. Designer uses the web browser to bring up the HTML display of data in the software-versioning feature (when such data needs to be displayed) as well as supporting the Actel-on-the-Web feature.

Starting other Applications from Designer

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

Note: This is supported on the PC only.

To add an application to the Tools menu:

1. **From the Tools menu, click *Customize*.** The customize dialog box is displayed as shown in Figure 2-50.

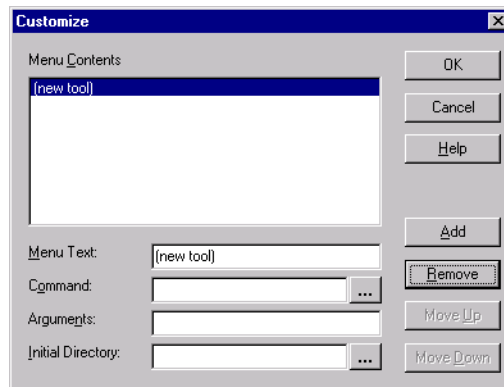


Figure 2-50. *Customize* Dialog Box

2. **Enter the application name in the Menu Text area.** This text appears 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 appear in the Tools menu in the same order as they do in the Menu Contents list box.

Saving a 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 you 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.

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 is displayed, as shown in Figure 2-51

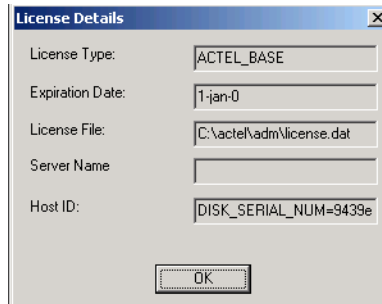


Figure 2-51. License Details Dialog Box

This information cannot be edited, it is for display purposes only.

Ending the Designer Session

To end a Designer session, choose the Exit command from the File menu. 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.

Scripting

The Designer software allows designers to run scripts in Tcl (Tool Command Language) for simple or complex tasks. You can run scripts from the Windows or UNIX command line or store and run a series of commands in a “.tcl” batch file.

This chapter contains information on using Tcl inside Designer. This chapter also contains a list of all the extension commands we have added to Tcl.

Tcl Overview

Tcl is a fairly simple programming language. If you have programmed before, you can learn enough to write interesting Tcl programs within a few hours. This chapter provides a quick overview of the main features of Tcl. After reading this overview, you'll probably be able to start writing simple Tcl scripts on your own; however, we recommend you consult one of the many available Tcl books for more complete information. You can also find Tcl information on the Web at various locations, such as <http://www.scriptics.com>.

Basic Syntax

Tcl scripts are made up of commands separated by new lines or semicolons. Commands all have the same basic form, such as:

```
expr 20 + 10
```

This command computes the sum of 20 and 10 and returns the result, 30.

Each Tcl command consists of one or more words separated by spaces. In this example there are four words: `expr`, `20`, `+`, and `10`. The first word is the name of a command and the other words are arguments to that command. All Tcl commands consist of words, but different commands treat their arguments differently. The `expr` command treats all of its arguments together as an arithmetic expression, computes the result of that expression, and returns the result as a string. In the `expr` command the division into words isn't significant: you could just as easily have invoked the same command as

```
expr 20+10
```

However, for most commands the word structure is important, with each word used for a distinct purpose. All Tcl commands return results. If a command has no meaningful result then it returns an empty string as its result.

Variables

Tcl allows you to store values in variables and use the values later in commands. The *set* command is used to write and read variables. For example, the following command modifies the variable *x* to hold the value *32*:

```
set x 32
```

The command returns the new value of the variable. You can read the value of a variable by invoking *set* with only a single argument:

```
set x
```

You don't need to declare variables in Tcl: a variable is created automatically the first time it is set. Tcl variables don't have types: any variable can hold any value.

To use the value of a variable in a command, use variable substitution as in the following example:

```
expr $x*3
```

When a *\$* appears in a command, Tcl treats the letters and digits following it as a variable name, and substitutes the value of the variable in place of the name. In this example, the actual argument received by the *expr* command is *32*3* (assuming that variable *x* was set as in the previous example). You can use variable substitution in any word of any command, or even multiple times within a word:

```
set cmd expr
```

```
set x 11
```

```
$cmd $x*$x
```

Command Substitution

You can also use the result of one command in an argument to another command. This is called command substitution:

```
set a 44
```

```
set b [expr $a*4]
```

When a `[` appears in a command, Tcl treats everything between it and the matching `]` as a nested Tcl command. Tcl evaluates the nested command and substitutes its result into the enclosing command in place of the bracketed text. In the example above the second argument of the second `set` command is `176`.

Quotes and Braces

Double-quotes allow you to specify words that contain spaces. For example, consider the following script:

```
set x 24
```

```
set y 18
```

```
set z "$x + $y is [expr $x + $y]"
```

After these three commands are evaluated variable `z` has the value `24 + 18 is 42`. Everything between the quotes is passed to the `set` command as a single word. Note that (a) command and variable substitutions are performed on the text between the quotes, and (b) the quotes themselves are not passed to the command. If the quotes were not present, the `set` command would have received 6 arguments, which would have caused an error.

Curly braces provide another way of grouping information into words. They are different from quotes in that no substitutions are performed on the text between the curly braces:

```
set z {$x + $y is [expr $x + $y]}
```

This command sets variable `z` to the value `"$x + $y is [expr $x + $y]"`.

Filenames

In Tcl-syntax, filenames must be enclosed in braces { } to avoid backslash-substitution, which is an issue with Windows-based filenames. The problem is that sequences of “\n” or “\t” are interpreted specially. Using the braces disables this special interpretation and specifies that the Tcl interpreter treat the enclosed string literally. Alternatively, double-backslash “\\n” and “\\t” would work as well as forward-slash directory-separators “/n” and “/t”.

For example, to specify a file on your Windows PC at c:\newfiles\thisfile.adb, use one of the following:

```
{C:\newfiles\thisfile.adb}
C:\\newfiles\\thisfile.adb
"C:\\newfiles\\thisfile.adb"
C:/newfiles/thisfile.adb
"C:/newfiles/thisfile.adb"
```

If there is white space in the filename-path, you should use the braces or double-quotes. For example C:\program data\thisfile.adb should be referenced in Tcl script as {C:\program data\thisfile.adb} or “C:\\program data\\thisfile.adb”.

Lastly, if you are attempting to use variables, you cannot use the braces { } since they by default turn off all special interpretation...including the dollar-sign. Instead do use either double-backslashes or forward slashes with double-quotes--something perhaps like this: “\$design_name.adb”.

Control Structures

Tcl provides a complete set of control structures including commands for conditional execution, looping, and procedures. Tcl control structures are just commands that take Tcl scripts as arguments. The example below creates a Tcl procedure called *power*, which raises a base to an integer power:

```
proc power {base p} {
    set result 1
    while {$p > 0} {
        set result [expr $result * $base]
        set p [expr $p - 1]
    }
    return $result
}
```


This script consists of a single command, *proc*. The *proc* command takes three arguments: the name of a procedure, a list of argument names, and the body of the procedure, which is a Tcl script. Note that everything between the curly brace at the end of the first line and the curly brace on the last line is passed verbatim to *proc* as a single argument. The *proc* command creates a new Tcl command named *power* that takes two arguments. You can then invoke *power* with commands such as:

```
power 2 6
```

```
power 1.15 5
```

When *power* is invoked, Tcl script evaluates the procedure body. While the body is executing it can access its arguments as variables: *base* holds the first argument and *p* holds the second.

The body of the *power* procedure contains three Tcl commands: *set*, *while*, and *return*. The *while* command does most of the work of the procedure. It takes two arguments, an expression ($\$p > 0$) and a body, which is another Tcl script. The *while* command evaluates its expression argument using rules similar to those of the C programming language and if the result is true (nonzero) then it evaluates the body as a Tcl script. It repeats this process over and over until eventually the expression evaluates to false (zero). In this case the body of the *while* command multiplied the result value by *base* and then decrements *p*. When *p* reaches zero the result contains the desired power of *base*. The *return* command causes the procedure to exit with the value of variable *result* as the procedure's result.

Command Origins

As you have seen, all of the interesting features in Tcl are represented by commands. Statements are commands, expressions are evaluated by executing commands, control structures are commands, and procedures are commands.

Tcl commands are created in three ways. One group of commands is provided by the Tcl interpreter itself. These commands are called built-in commands. They include all of the commands you have seen so far and many more (see below). The built-in commands are present in all Tcl applications.

The second group of commands is created using the Tcl extension mechanism. Tcl provides APIs that allow you to create a new command by writing a command procedure in C or C++ that implements the command. You then

register the command procedure with the Tcl interpreter by telling Tcl the name of the command that the procedure implements. In the future, whenever that particular name is used for a Tcl command, Tcl calls your command procedure to execute the command. The built-in commands are also implemented using this same extension mechanism; their command procedures are simply part of the Tcl library.

When Tcl is used inside an application, the application incorporates its key features into Tcl using the extension mechanism. Thus the set of available Tcl commands varies from application to application. The third group of commands consists of procedures created with the `proc` command, such as the power command created above. Typically, extensions are used for lower-level functions where C programming is convenient, and procedures are used for higher-level functions where it is easier to write in Tcl.

Other Features

Tcl contains many other commands besides the ones used in the preceding examples. Here is a sampler of some of the features provided by the built-in Tcl commands:

- More control structures, such as *if*, *for*, *foreach*, and *switch*.
- String manipulation, including a powerful regular expression matching facility. Arbitrary-length strings can be passed around and manipulated just as easily as numbers.
- I/O, including files on disk, network sockets, and devices such as serial ports. Tcl provides particularly simple facilities for socket communication over the Internet.
- File management: Tcl provides several commands for manipulating file names, reading and writing file attributes, copying files, deleting files, creating directories, and so on.
- Subprocess invocation: you can run other applications with the `exec` command and communicate with them while they run.
- Lists: Tcl makes it easy to create collections of values (lists) and manipulate them in a variety of ways.
- Arrays: you can create structured values consisting of name-value pairs with arbitrary string values for the names and values.
- Time and date manipulation.

- Events: Tcl allows scripts to wait for certain events to occur, such as an elapsed time or the availability of input data on a network socket.

Tcl Extension Commands Added by Designer

In the descriptions below, optional arguments are enclosed in square brackets [] for notational purposes only. Also notational-only, valid values are displayed within enclosing parentheses (), separated by bars |.

backannotate

This is equivalent to executing the Back-Annotate command within the Tools menu (see [“Back-Annotation” on page 58](#)).

Syntax:

```
backannotate
```

```
backannotate -name name -format type -language simlang [-dir dir] [-netlist] [-pin]
```

close_design

This is equivalent to executing a Close command within the File menu.

Syntax:

```
close_design
```

compile

This is equivalent to executing a Compile command within the Tools menu (see [“Compiling a Design” on page 44](#)).

Syntax:

```
compile [-nl_pins_overwrite]
```

export

This is equivalent to executing a command on the Export sub-menu within the File menu (see [“Exporting Files” on page 67](#)).

Syntax:

```
export -format edif \  
    -edif_flavor ( generic | viewlogic | mgc | orcad | workview  
    ) \  
    {filename}  
  
export -format ( afm | dio | fus ) [-signature value] {file-  
name}  
  
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

Syntax:

```
get_defvar variable
```

import

Import Source Files and Import Auxillary Files are new Designer enhancements. However, the previous Tcl scripts for import are still supported.

```
import -edif_flavor ( generic | viewlogic | mgc | orcad | work-  
view ) \  
    {filename}  
  
import -format vhd1 -top_entity entity_name [-vhd187] {file-  
name}  
  
import -format adl -netlist_naming (generic | verilog | vhd1 )  
\  
    {filename}  
  
import -format ( crt | dcf | verilog | vhd1 ) {filename}
```

import_aux

This is the same as executing the Import Auxiliary Files command from the File menu.

```
import_aux -format "crt" \  
{D:\testarea\designs\anita\test.crt}
```

import_source

This is equivalent to executing an Import Source File command within the File menu (see [“Importing Source Files” on page 27](#)).

Syntax:

```
import_source -format "edif" -edif_flavor "GENERIC" \  
{D:\testarea\designs\anita\test.edn} -format "pin" \  
{D:\testarea\designs\anita\test1.pin}
```

layout

This is equivalent to executing a Layout command within the Tools menu (see [“Layout” on page 50](#)).

Syntax:

```
layout [-timing_driven] [-incremental ( on | off | fix )]
```

layout (advanced options for ONO families)

This is equivalent to executing commands within the Advanced Layout Options dialog box.

Syntax:

```
layout [-timing_driven] [-incremental inc_mode] [-extended_run  
ext_mode]  
where inc_mode:= "on" | "off" | "fix" , ext_mode:= "on" |  
"off"
```

layout (advanced options for the SX family)

This is equivalent to executing commands within the Advanced Layout Options dialog box.

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

This is equivalent to executing a New command within the File menu (see [“Starting a New Design” on page 13](#)).

Syntax:

```
new_design -name design_name -family family_name
```

open_design

This is equivalent to executing an Open command within the File menu (see [“Opening an Existing Design” on page 14](#)).

Syntax:

```
open_design {filename}
```

report

This is equivalent to executing a Reports command within the Tools menu (see [“Generating Reports” on page 72](#)).

Syntax:

```
report -type flipflop [-fmt ( summary | extended )] {filename}
```

```
report -type pin [ -listby (name | number) ] {filename}
```

```
report -type status {filename}
```

```
report -type timing [-sortby ( actual )] [-maxpaths number]
```

```
[-breakclkpin] [-breakclrpip] {filename}
```

save_design

This is equivalent to executing a Save command within the File menu (see “Saving a Design” on page 87).

Syntax:

```
save_design  
  
save_design {filename}
```

set_defvar

Syntax:

```
set_defvar variable value
```

Note: Specifying “NULL” as the value unsets the variable.

set_design

This is equivalent to executing a Setup Design command within the Tools menu (see “Changing Design Name and Family” on page 65).

Syntax:

```
set_design -name design_name -family family_name
```

set_device

This is equivalent to using the Device Selection Wizard.

Syntax:

```
set_device \  
-family family_name \  
-die die_name \  
-package package_name \  
-speed speed_grade \  
-voltage voltage \  
-voltrange voltrange \  
-temprange temprange \  
-pci ( yes | no ) \  
-jtag ( yes | no ) \  

```

```
-probe ( yes | no ) \  
-itol ( 3.3 | 5.0 ) \  
-io_trip ( pci | ttl ) \  
-trst ( yes | no )
```

PDC Commands

reset_iobank

```
reset_iobank <bankname>
```

Resets the I/O bank's technology. This may result in all or a few ports in the bank to be unplaced.

reset_io command

```
reset_io <portname>
```

portname ::= <single port name> | <regexp>

<regexp> ::= *

All parameters are required. The port is assigned the default I/O technology of the design. This default is read from the default system and can also be changed using the Design Setup Wizard.

reset_net_criticality

```
reset_net_critical hier_net_name {hier_net_name}
```

hier_net_name ::= <AFL netname> | <net_regexp>

<net_regexp> ::= A limited CShell like regular expression using ?, *, [and] characters,

Reset the criticality of the net to the default (5). The command must have at least 1 net name parameter. The Net names are AFL names.

set_io

```
set_io <portname> -standard <io_std> -slew <high | low> -  
strength <8 | 12 |16 |24> -delay < on | off > -register <on |  
off> -pinname <pinname> -fixed <yes | no> -bank <bankname>
```


io_std ::= lvttl | pci | pcix | lvcmos25 | lvcmos18 | lvcmos15 | hstl1 |
sstl31 | sstl32 | sstl21 | sstl22 | gtlp33 | gtlp25

on_off ::= on | off

portname ::= <single port name> | <regex>

strength = output drive strength. Applicable for outputs only.

<regex> ::= *

<pinname> ::= a valid package pin name (either a number or alphanumeric)

All parameters except the port name are optional. If a parameter is not specified, the value will not be changed as long as it is consistent with other settings. For example, setting the I/O standard to PCI forces the slew to High. Port names are the original (i.e. no naming translations are done) names from the imported netlist. If a port is not specified in the PDC file, its settings are not changed.

Note: Assigning an I/O standard to a port may invalidate its location. In this case, it is unplaced automatically.

set_iobank

This command specifies the I/O bank's technology to the given standard. This may result in many ports in the bank to be unplaced. I/O Bank name and the All parameters are required.

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

bankname = bank<0 .. 7>

vcci_voltage = 3.3, 2.5, 1.8

vccr_voltage = 1.5, 2.5,...

-inputdelay <delay_value> = Enables the input delay. Delay value can be from 1-31.

-lpininput <On|Off> = Enables or disables the Low Power Mode for input buffers)

-lpoutput <On|Off> = Enables or disables the LowPowerMode for output buffers)

set_location

```
set_location <clustername> x y
```

clustername ::= <module instance name> in the netlist

x, y ::= module instance name coordinates

set_net_criticality

```
set_net_critical <criticality_number> hier_net_name  
[hier_net_name]
```

criticality_number ::= 1..10

hier_net_name ::= <AFL netname> | <net_regexp>

<AFL netname>::=

<net_regexp> ::= A limited CShell like regular expression using ?, *, [and] characters,

Net criticality is used to influence placement and routing in favor of performance. The allowable range of net criticality is 1-10.

Set the criticality of the net to the given number. The command must have at least 2 parameters. (in the above order) The net names are AFL names, meaning they must be visible in Timer and ChipEdit (note: Default criticality is 5).

set_vref

```
set_vref {-bank <bankname>} <pinnum> }<pinnum>}
```

bankname is one of Bank0, Bank1 Bank7

<pinnum> is the alphanumeric pinname

Sets the given pins as Vref pins. The bank name is optional. It is an error to give bank name and mention pins that do not belong to that bank. Pins that do not belong to a bank that needs a Vref are ignored.

set_vref_defaults

```
set_vref_defaults -bank <bankname>
```

bankname is one of Bank0, Bank1 Bank7

Sets the default vref pins for the given bank. This command is ignored if the bank does not need a vref (because of the I/O technology it supports)

Error Handling

After executing each of these commands, a valid TCL result is set so you can check the status. The result is a string value and can be one of these values.

“OK”

“ERROR: Unknown port”

“ERROR: Unknown I/O Standard”

“ERROR: Read only I/O Standard”

“ERROR: Error in setting I/O Standard”

“ERROR: Unknown I/O Attribute”

“ERROR: I/O Attribute is not applicable”

“ERROR: Read only I/O Attribute”

“ERROR: Invalid Package pin”

“ERROR: Illegal or Invalid assignment to Package pin”

“ERROR: Not implemented yet”

“ERROR: Net criticality must be 1-10”

“Error: Unknown Net”

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

Sample PDC Script

```
set a [set_io reg_out<3> -iostd LVTTTL]

if {$a != "OK"} {

    puts "ERROR: $a"

}

set b [set_iobank bank3 -iostd LVTTTL]
```

```
if {$b != "OK"} {  
  
    puts "set_iobank: $b"  
  
}
```

SDC Commands

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 Synopsis Design Compiler, Prime Time, and Synplicity tools can generate SDC descriptions or the user can generate the SDC file manually.

Design Object Access Commands

Most constraint commands require a command argument. Designer supports the SDC access commands shown in Table 3-1.

Table 3-1. Supported SDC Access Commands

Design Object	Access Command
Clock	get_clocks
Port	get_ports

get_clocks

This command 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 timing constraint commands in Table 3-2:

Table 3-2. Supported SDC Timing Constraint Commands

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.

Supported arguments:

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

The following is a description of command syntax for specifying a clock:

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

Note:

- *period_value* is mandatory and must be specified in ns. No clock is created if the period is not supplied.
- *clock_name* is optional if *port_pin_list* is supplied.

- *edge_list* is optional. If supplied, it has to contain exactly 2 edges to be taken into account. The duty cycle information is 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 11 -name CK1 -waveform {0 2 5 7} (valid, but the waveform is ignored.)

create_clock -period 6 [get_ports CK1]

create_clock -period 2 -name CLOCK [get_ports ck1] (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}] (There is more than one name in the port_pin_list.)

create_clock -period 7 -name ck [get_ports {clk1 clk2}]

set_max_delay

Supported Arguments:

[-from *from_list*]

[-to *to_list*]

delay_value

Note:

- *from_list* is mandatory.
- *to_list* is mandatory.

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.)

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.

Object Access Commands: Only *get_ports* and *get_clocks* are supported.

Timing Constraints: Only *create_clock* and *set_max_delay* are supported.

Tcl Commands for Timer

timer_remove_all_constraints

This is equivalent to executing the Remove All Constraints command in the Edit menu.

timer_commit

This is equivalent to executing the Commit command in the File menu. This is needed after all Timer commands to save changes made in Timer.

timer_restore

This restores previous committed constraints.

timer_set_maxdelay

This is equivalent to adding a maximum delay constraint in the Path/Set grid.

Syntax:

```
timer_set_maxdelay -from "pin1" -to "pin2" -unit "ns|ps"  
-delay delay
```

timer_get_maxdelay

This occurs at the initiation of Timer. All maximum delay constraints are displayed in the set grid.

Syntax:

```
timer_get_maxdelay -from "pin1" -to "pin2"
```

timer_setenv_clock_freq

This is equivalent to setting a required clock frequency in the Clocks tab (in Mhz).

Syntax:

```
timer_setenv_clock_freq -clock "name" -freq frequency
```

timer_setenv_clock_period

This is equivalent to setting required clock period in the Clocks tab (in ns).

Syntax:

```
timer_setenv_clock_period -clock "name" -unit "ns|ps" -period  
period
```

timer_add_clock_exception

This is equivalent to adding a clock exception in the Clocks tab.

Syntax:

```
timer_add_clock_exception -clock "name" -pin "pin" -dir ( from  
| to )
```

timer_remove_clock_exception

This is equivalent to removing a clock exception in the Clocks tab.

Syntax:

```
timer_remove_clock_exception -clock "name" -pin "pin" -dir
```


(from | to)

timer_add_stop

This is equivalent to adding stop points on the Breaks tab.

Syntax:

```
timer_add_stop -pin "pin"
```

timer_add_pass

This is equivalent to adding a pass pin on the Breaks tab.

```
timer_add_pass -pin "pin"
```

timer_remove_stop

This is equivalent to removing a stop point on the Breaks tab.

Syntax:

```
timer_remove_stop -pin "pin"
```

timer_remove_pass

Syntax:

```
timer_remove_pass -pin "pin"
```

timer_get_path_constraints

This is equivalent to displaying all the paths that have a max delay in the set grid.

timer_get_clock_constraints

This is equivalent to displaying the required clock frequency (clock constraint) for the clock input.

Syntax:

```
timer_get_clock_constraints -clock "name"
```

timer_get_clock_actuals

This is equivalent to finding the actual clock frequency. Occurs when timer is initiated.

Syntax:

```
timer_get_clock_actuals -clock "name"
```

timer_get_path

This is equivalent to clicking on a path in the Set grid to display paths in the Paths grid.

Syntax:

```
timer_get_path -from "pin1" -to "pin2"
               -exp ( no | yes )
               -sort ( actual | slack )
               -order ( long | short )
               -case ( worst | typ | best )
               -maxpath number
               -maxexpath number
               -mindelay delay
               -maxdelay delay
               -breakatclk ( no | yes )
               -breakatclr ( yes | no )
```

Tcl Commands for PinEdit

pin_assign

This is equivalent to assigning a pin. This assigns the pin, but does not fix its assignment.

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 for a given port:

```
-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.

pin_commit

This is equivalent to committing all changes in PinEdit. This is needed after all pin commands to save changes.

Syntax:

pin_commit

pin_fix

This is equivalent to fixing a pin assignment.

Syntax:

pin_fix -port <portname>

pin_fix_all

pin_unassign

This is equivalent to unassigning a pin.

Syntax:

pin_unassign -port <portname>

pin_unassign_all

pin_unfix

This is equivalent to unfixing a pin assignment.

Syntax:

```
pin_unfix -port <portname>
```

```
pin_unfix_all
```

Running Scripts from the Command Line

To execute a Tcl script file in Designer from the command line, type the following at the command line prompt:

```
<location of Actel software>/bin/designer script:<filename>
```

where:

<location of Actel software> is the root directory in which you installed the Actel software.

<filename> is the name, including a relative or full path, of the Tcl script file you wish to execute.

To pass arguments from the command line to your Tcl script file, change the script:<filename> to script:"<filename arg1 arg2 ...>"

where:

arg1, arg2 and ... are arguments you are passing to the script file.

Note: The quotes are required when passing arguments to the Tcl script.

To access arguments inside the Tcl script, use \$argv0 to access the Tcl script filename.

Use the following command to access arguments from the command line:

```
[lindex $argv 0]
```

Note: \$argv0 is not the same as [index \$argv 0]. Also, the number in the previous command is the position of the argument on the command line you are trying to access. Counting starts at 0.

Running Scripts within Designer

To execute a Tcl script file within Designer:

1. **In the File menu, click *Execute Script File*.** This displays the Execute Script dialog box, as shown in Figure 3-1.

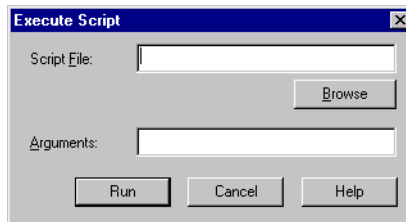


Figure 3-1. Execute Script Dialog Box

2. **In the Script File edit box, enter the name of the script file.** You can use the browse button to bring up a file dialog browser and select the script file from that dialog box.
3. **Enter arguments you wish to pass to your Tcl script in the Arguments edit box.** For information about accessing arguments passed to a Tcl script, see page 113.
4. **Click *Run*.**

Recording Scripts

Designer can export a Tcl script file that contains commands executed in the current session. You can then use this exported Tcl script to re-execute the same commands interactively or in batch. You can also use this exported script to become more familiar with Tcl syntax.

To export a Tcl script from Designer:

1. **In the File menu, click Script Files from the Export sub-menu.** The Export Tcl Script File dialog box is displayed, as shown in Figure 3-2.



Figure 3-2. Export Tcl Script Files Dialog Box

2. **Specify a filename.**
3. **Click Save.** A Script Export Options dialog box is displayed, as shown in Figure 3-3.

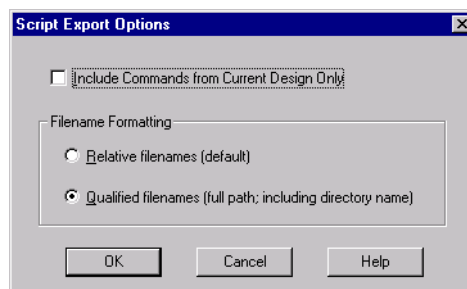


Figure 3-3. Script Export Options Dialog Box.

4. **Check the Include Commands for Current Design Only checkbox if you want to export commands relative to the current design only.** This only applies if you had open more than one

design in your current session. If so, and you do not check this box, Designer exports all commands from your current session.

5. Select the radio box for the appropriate filename formatting.

If you want to export filenames relative to the current working directory, select relative filenames formatting. If you want to export filenames that include a fully specified path, select qualified filenames formatting.

6. Click OK. The Tcl script file is exported to the specified filename.

Note: When exporting Tcl scripts, Designer always encloses filenames in curly braces since this is portable and there are no variables used in the generated Tcl commands.

Example Scripts

Basic Design Flow

```
#Set up a new design
new_design -name "prepi" -family "54SXA"

#set device name, package name
set_device -die "A54SX32A" -package "176 TQFP"

#set device speed and operating conditions
set_device -speed "-1" -temprange "com" -voltrange "com"

#import netlist and pin files
import -format "edif" -netlist_naming "Generic" \
      -edif_flavor "GENERIC" {prepi.edn}
import -format "pin" {prepi.pin}

compile

#layout standard mode
layout -incremental "OFF"

#extract sdf file
export -format "sdf" {prepi.sdf}

save_design {prepi.adb}

close_design
```


Command Line Arguments to the Script File

```
puts "Script name is $argv0" ; # accessing the scriptname
puts "first argument is [lindex $argv 0]"
puts "second argument is [lindex $argv 1]"
puts "third argument is [lindex $argv 2]"
puts "number of argument is [llength $argv]"

set des_name [lindex $argv 0]
puts "Design name is $des_name"
```

Note: The comment delimiter, #, must be the first character on a line or the first character following a semicolon (in Tcl, commands are separated by new lines or semicolons).

Exception Handling

If you want to control the flow of the Designer software based on certain conditions (e.g. success or failure of certain commands), you can use the tcl built-in catch command as follows:

```
if { [ catch {open_design $des_name.adb} ] } {

    puts "Cannot open $des_name.adb"

    export -format "log" -diagnostic $des_name.log"
    exit 1
} else {

    puts "Design $des_name.adb Successfully Opened"
}

## set layout mode to standard
layout -incremental "OFF"

if { [ catch {layout} ] } {

    puts "Layout Failed"

    export -format "log" -diagnostic $des_name.log"
    exit 1
} else {

    puts "layout successful"
```

```
export -format log "$des_name.log"  
  
save_design "$des_name.adb";  
close_design  
}
```

Extended Layout

These extended layout Tcl scripts force layout to run with an extended set of parameters. These scripts cause the layout to run for a much longer period of time in an attempt to achieve greater performance and routability.

The purpose of these scripts is to run multiple layout iterations on a design using different initial seeds for the placer. Each script checks the timing report for the best performance from all the runs, and saves that design.

The design must be completed through Compile.

Defaults: Run 5 Layout iterations with different placer seeds. Layout is done in timing driven mode. Performance is measured using the default mode (highest frequency of the slowest clock).

Arguments:

`-n value`

allows you to specify the number of iterations of layout. The maximum is 25 and the default value is 5.

`-c clockname`

allows you to specify the clock that will be optimized for the maximum register to register delay.

Results:

With no clockname provided:

The iteration with the highest frequency of the slowest clock is chosen as the best result. The design is saved with this result.

With clockname provided:

The iteration with the minimum value of the maximum register to register delay for the specified clock is chosen as the best result. The design is saved with this result.

Running from the Designer GUI

Recommended for Windows users.

To run the script:

- 1. From the File menu, click *Execute Script*.**
- 2. From the Execute Script dialog, use the file browser to select the *iterate.tcl* script found in the script sub-directory under the software installation directory.**
- 3. Enter any of the optional parameters in the "Arguments" field.**
- 4. Click the *Run* button.**

Running from the Command Line

Recommended for UNIX users.

Additional Arguments:

```
-adb filename
```

Specify the design file (.adb) on which to run the extended layout script.

To run the script:

- 1. From your command line shell, execute the following:**

```
<location of Actel software>/bin/acttclsh <location of Actel software>/scripts/sh_iterate.tcl -adb filename [-n value ] [-c clockname]
```

- 2. The script runs the Designer software in batch mode and report the results in the shell window.**
- 3. On error, a non-zero value will be returned.**

Axcelerator Script

```
#Set up a new design
new_design -name "multiclck" -family "Axcelerator" -path {..}
# Set device, package, speed grade, default I/O standard and
operating conditions
set_device -die "AX1000" -package "BG729" -speed "-3" -voltage
"1.5" -iostd "LVTTTL" -temprange "COM" -voltrange "COM"
# Import the netlist
import -format "verilog" {multiclck.v}
# Compile the netlist
compile
# Import a PDC file
import_aux -format "pdc" {multiclck.pdc}
# Run standard layout
layout -incremental "OFF"
# Generate backannotated sdf and netlist file
backannotate -name {multiclck_ba} -format "sdf" -language
"Verilog"
# Generate timing report
report -type "timing" -sortby "actual" -maxpaths "100"
{report_timing.txt}
# Generate programming file
export -format "AFM" -signature "ffff" {multiclck.afm}
```

Constraints in ProASIC and ProASIC^{PLUS} Devices

This appendix describes the constraints that can be used to guide place and route, and to set optimization criteria for ProASIC and ProASIC^{PLUS} devices. These constraints may be the forward timing SDF file generated by the synthesis tool. If the constraints are relative to the placement, the global resources and the netlist optimization, they should be included in a file with the .gcf extension.

ProASIC constraint files (GCF), which are imported using Designer, must use the language and syntax described in this appendix.

Types of Constraints

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

- Timing constraints
- Global resource constraints
- Netlist optimization constraints
- Placement constraints
- I/O constraints

ProASIC Timing Constraints

Timing constraints are used to ensure that a design meets the required timing performance. Constraints can be entered using an 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.

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 backannotate 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.

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.

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 importer generates internal path constraints for 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 will not be considered in 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_register_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_*;
```

Global Resource Constraints

Each ProASIC 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 global 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.

Global Resource Constraint Syntax

The following section describes the global resource constraints that can be set in your .gcf file.

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, the checker 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 the checker program will consider 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.

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;
```

use_global

Use this statement to guide place-and-route to route the net using the specified global spine from the global network.

```
use_global spine netname;
```

Where “spine” is one of the spines T1 to T<n> or B1 to B<n>, “netname” is the name of the net. See table for a summary of available spines.

Table A-1. Global Spine Usage^a

Device	Spine
A500K050	T1 to T3
	B1 to B3
A500K130	T1 to T5
	B1 to B5

Table A-1. Global Spine Usage^a

Device	Spine
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

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

This statement instructs the placer to place the driver cell of the net close to the indicated spine and place all other cell instances connected to that net within the spine region. Then the router will be instructed to route this net through the specified global spine resource.

For Example, the following statement specifies that the net u3/u1/clk should be routed using the global spine B3.

```
use_global B3 u3/u1/clk;
```

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* 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.

The following sections describe the netlist optimization constraints that are available.

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.

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

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_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 global IOs are excluded from the region constraints.

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

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.

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

use_global

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

```
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<=m will mean B_n, B_{n+1}, ... B_m
2. T_n, T_m : n<=m will mean T_n, T_{n+1}, ... T_m
3. B_n, T_m : n<=m will mean B_n, T_n, B_{n+1},T_{n+1} ... B_m, T_m
4. T_n, B_m : n<=m will mean B_n, T_n, B_{n+1},T_{n+1} ... B_m, T_m

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/*;
```

The statement in this example will enable only the buffer and inverter optimization types and optimization will be done on all instances except those contained in the block called /U1/myblock.

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 kept 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.

Creating the I/O Allocation Pin Map

If the printed circuit board (PCB) has been designed that will include the ProASIC part, there may be a need to fix the pin assignments for the physical design. The easiest way to do this is to use PinEdit. Refer to the *PinEdit User's Guide* for more information.

Package Pin and Pad Location

Generally, users will be concerned with the mapping of signals (ports) to the pins of the selected package. However, users 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.

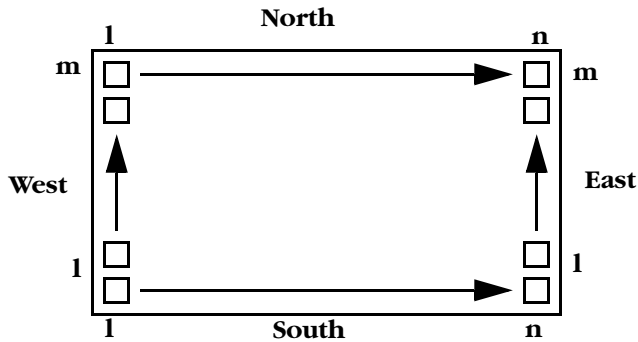


Figure A-1. Pad Locations

Individual Placement Constraint Syntax

This section describes the types of individual placement constraints that are available. Individual placement constraints can be useful if users want to force certain critical blocks to be fixed locations, or guide the placement program for the initial location of blocks.

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); //dont.use 33% tile on East

Side
```

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_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; 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 (required if no package pin or side is given). “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 A11 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_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 { package_pin_name | location_definition };
```

For example:

```
set_io A9 in1;
```

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

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_empty_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_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;
```

Macro Placement Constraint Syntax

This section describes the types of macro placement constraints that are available. Macro placement constraints can be useful if users want to reuse a previously produced placement for a subcircuit of a design or if the design is using a predefined core. Use this statement to define the locations of a sub-design as a macro so that users are able to reuse this placement in different instantiations of the sub-design. The macro is defined in terms of individual core placements. The hierarchical instance names, appearing in these constraints, are considering the macro as the top of the design.

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;
```


Constraint Quick Reference

- “create_clock” on page 122
- “generate_paths” on page 123
- “set_input_to_register_delay” on page 124
- “net_critical_ports” on page 125
- “set_critical” on page 125
- “set_critical_port” on page 126
- “set_max_path_delay” on page 126
- “set_switch_threshold” on page 127
- “set_auto_global” on page 129
- “set_global” on page 129
- “dont_fix_globals” on page 130
- “use_global” on page 130
- “dont_optimize” on page 133
- “optimize” on page 133
- “set_net_region” on page 133
- “set_max_fanout” on page 134
- “dont_touch” on page 134
- “set_empty_location” on page 136
- “set_empty_io” on page 137
- “set_initial_io” on page 137
- “set_io” on page 138
- “set_location” on page 138
- “set_initial_location” on page 139

Constraint File Syntax Summary

This section summarizes the syntax used in the constraint file format.

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.

Table A-2. Syntax Conventions for Notation

Notation	Description
<i>item</i>	Represents a syntax item.
<i>item</i> ::= definition	<i>item</i> is defined as definition.
<i>item</i> ::= <i>definition1</i> = <i>definition2</i>	<i>item</i> is defined as either <i>definition1</i> or <i>definition2</i> . (Multiple alternative syntax definitions are allowed.)
[<i>item</i>]	<i>item</i> is optional.
{ <i>item</i> }	<i>item</i> 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.

Table A-3. 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.

**Syntax
Summary**

```

constraint_file           ::= constraint_statements
constraint_statements   ::= {constraint_statement}
constraint_statement     ::= critical_port_statement
                        | dont_fix_statement
                        | dont_optimize_statement
                        | dont_touch_statement
                        | empty_io_statement
                        | empty_location_statement
                        | initial_location_statement
    
```

	<i>io_statement</i>
	<i>jtag_statement</i>
	<i>location_statement</i>
	<i>macro_define_statement</i>
	<i>max_fanout_statement</i>
	<i>net_criticality_statement</i>
	<i>net_critical_ports_statement</i>
	<i>net_delay_statement</i>
	<i>optimize_statement</i>
	<i>path_delay_statement</i>
	<i>read_statement</i>
	<i>timing_fanout_statement</i>
<i>create_clock_statement</i>	::= CREATE_CLOCK -PERIOD POSFLOAT { <i>net_name</i> <i>port_name</i> }
<i>critical_port_statement</i>	::= SET_CRITICAL_PORT <i>criticality_number</i> <i>hier_inst_names</i> ;
<i>dont_fix_statement</i>	::= DONT_FIX_GLOBALS ;
<i>dont_optimize_statement</i>	::= DONT_OPTIMIZE [{ <i>buffer inverter clocktree</i> <i>resettree const dangling</i> }];
<i>dont_touch_statement</i>	::= DONT_TOUCH <i>hier_net_names</i> ;
<i>empty_io_statement</i>	::= SET_EMPTY_IO <i>pad_location_defs</i> ;
<i>empty_location_statement</i>	::= SET_EMPTY_LOCATION <i>location_def</i> ; SET_EMPTY_LOCATION <i>location_region</i> ;
<i>generate_paths_statement</i>	::= GENERATE_PATHS [- COVER_DESIGN] [- TOP POSNUMBER] [- MAX_PATHS POSNUMBER];
<i>global_net_statement</i>	::= { SET_GLOBAL SET_NOGLOBAL } <i>hier_net_names</i> ;

<i>initial_io_statement</i>	::= SET_INITIAL_IO <i>io_location_def</i> <i>io_port_names</i> ;
<i>initial_location_statement</i>	::= SET_INITIAL_LOCATION <i>location_def</i> <i>hier_inst_names</i> ; SET_INITIAL_LOCATION <i>location_def</i> <i>hier_subdesign_inst_name</i> <i>macro_name</i> [<i>transformations</i>];
<i>io_statement</i>	::= SET_IO <i>io_location_def</i> <i>io_port_names</i> ;
<i>location_statement</i>	::= SET_LOCATION <i>location_def</i> <i>hier_inst_name</i> ; SET_LOCATION <i>location_region</i> <i>hier_name_wildcard</i> ; SET_LOCATION <i>location_def</i> <i>hier_subdesign_inst_name</i> <i>macro_name</i> [<i>transformations</i>];
<i>macro_define_statement</i>	::= MACRO <i>macro_name</i> <i>macro_location_def</i> { <i>macro_statements</i> }
<i>macro_statements</i>	::= { <i>macro_statement</i> };
<i>macro_statement</i>	::= SET_LOCATION <i>location_def</i> <i>hier_inst_name</i> ; SET_INITIAL_LOCATION <i>location_def</i> <i>hier_inst_name</i> ; SET_EMPTY_LOCATION <i>location_def</i> ;
<i>max_fanout_statement</i>	::= SET_MAX_FANOUT POSNUMBER;
<i>net_criticality_statement</i>	::= SET_CRITICAL <i>criticality_number</i> <i>hier_net_names</i> ;
<i>net_critical_ports_statement</i>	::= NET_CRITICAL_PORTS <i>hier_net_name</i> <i>critical_ports</i> ;
<i>optimize_statement</i>	::= OPTIMIZE [{ <i>buffer inverter resettree clocktree const</i> <i>dangling</i> }];
<i>path_delay_statement</i>	::= SET_MAX_PATH_DELAY <i>delay_value</i> <i>delay_path</i> ;
<i>read_statement</i>	::= READ [- FORMAT <i>format</i>] [- ECO] <i>file</i> ;
<i>set_false_path_statement</i>	::= SET_FALSE_PATH [- FROM <i>portname</i>]

```

                                [-THROUGH portname] [-TO portname]
set_input_to_register_delay_statement ::= SET_INPUT_TO_REGISTER_DELAY
                                        delay_value [-FROM port_name];
set_multicycle_path_statement ::= SET_MULTICYCLE_PATH num_cycles [-
                                FROM port_name] [-TO port_name] [-
                                THROUGH port_name];
set_register_to_output_delay_statement ::=
                                SET_REGISTER_TO_OUTPUT_DELAY
                                        delay_value [-TO port_name]
set_default_switch_threshold_statement ::=
                                SET_DEFAULT_SWITCH_THRESHOLD
                                POSNUMBER;

set_switch_threshold_statement ::= SET_SWITCH_THRESHOLD
                                POSNUMBER hier_net_name;
use_global_statement ::= USE_GLOBAL global_region hier_net_name;
criticality_number ::= POSNUMBER
critical_ports ::= instance_port_name [, instance_port_name]
delay_path ::= instance_port_name [, instance_port_name]
delay_value ::= POSFLOAT
file ::= FILEIDENTIFIER
global_region ::= T1 to T<n> | B1 to B<n>
format ::= SDF | GF
hier_inst_name ::= QPATH
hier_inst_names ::= hier_inst_name [, hier_inst_name]
hier_net_name ::= QPATH
hier_net_names ::= hier_net_name [, hier_net_name]
hier_subdesign_inst_name ::= QPATH
instance_port_name ::= hier_inst_name.port_name

```

<i>io_location_def</i>	::= <i>side</i> <i>pad_location</i> <i>package_pin</i>
<i>io_port_names</i>	::= <i>io_port_name</i> [, <i>io_port_name</i>]
<i>io_port_name</i>	::= IDENTIFIER
<i>location_def</i>	::= (<i>x</i> , <i>y</i>)
<i>location_region</i>	::= (<i>x1</i> , <i>y1</i> x2, <i>y2</i>)
<i>transformation</i>	::= flip lr flip ud rotate 90 cw rotate 180 cw rotate 270 cw rotate 90 ccw rotate 180 ccw rotate 270 ccw
<i>macro_location_def</i>	::= (<i>x1</i> , <i>y1</i> x2, <i>y2</i>)
<i>macro_name</i>	::= IDENTIFIER
<i>offset</i>	::= POSNUMBER
<i>package_pin</i>	::= IDENTIFIER
<i>pad_location_defs</i>	::= <i>pad_location_def</i> [, <i>pad_location_def</i>]
<i>pad_location_def</i>	::= <i>package_pin</i> <i>pad_location</i>
<i>pad_location</i>	::= (<i>side</i> , <i>offset</i>)
<i>port_name</i>	::= IDENTIFIER
<i>side</i>	::= N S E W
<i>transformations</i>	::= <i>transformation</i> transformation transformations
<i>xgrid</i>	::= POSNUMBER
<i>ygrid</i>	::= POSNUMBER

Setting Up a Printer in UNIX

This appendix contains instructions on how to set-up a printer on Unix.

To specify available printers:

1. **In the File menu, click Print.** This displays the Print dialog box (Figure B-1).

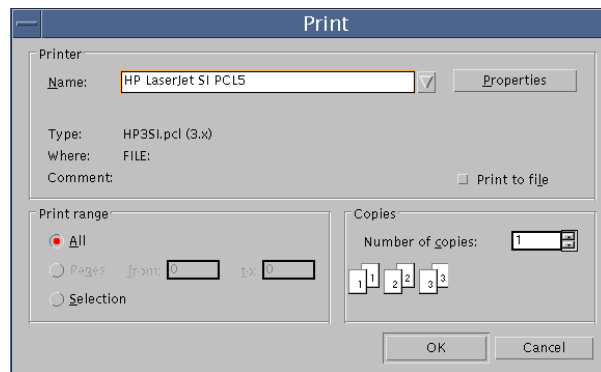


Figure B-1. Print Dialog Box

2. **Click Properties.** This displays the Printer Setup dialog box (Figure B-2).

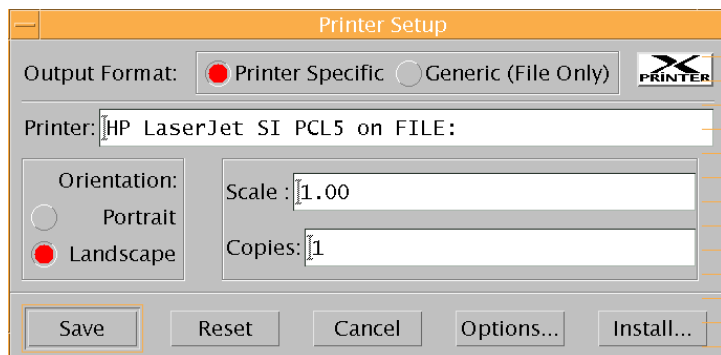


Figure B-2. Printer Setup Dialog Box

3. **Click *Install*.** This displays the Printer Installation dialog box (Figure B-3).

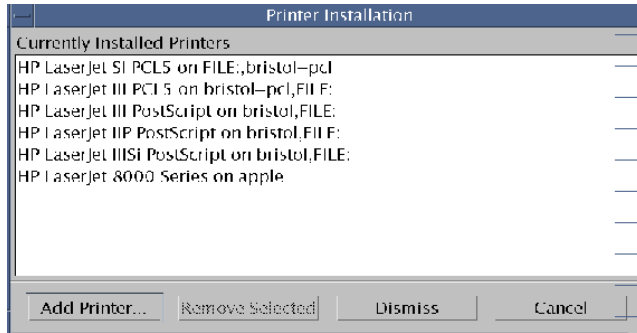


Figure B-3. Printer Installation Dialog Box

4. **Click *Add Printer*.** This displays the Add Printer dialog box (Figure B-4).

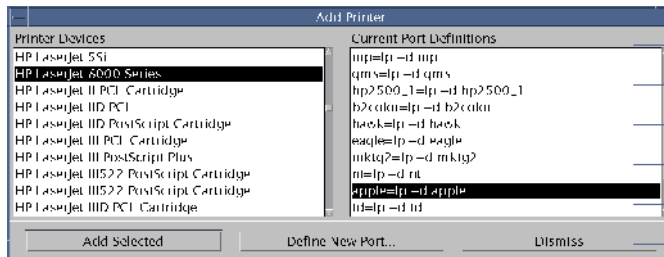


Figure B-4. Add Printer Dialog Box

5. **Specify the printer command by defining a new port.**

The command used to send output to a specific printer depends on the platform, printer, and how the printer is connected to your system. For example, if a printer is connected directly to your system, the following might be valid a print command:

lp -d ps

If your printer is connected to a different system on your network, your printer command will specify how to connect to that system. For example,

if a printer is connected to the system *bandit* on your network, the following might be a valid print command:

rsh bandit lp -d ps

A printer port is an alias for the print command.

To define a new port, click Define New Port. This displays the Ports dialog box.

- 6. Type the port definition in the Edit Port edit box.** For example, suppose you have two printers: ORION and SIRIUS. Your port definitions may look like the following examples:

```
ORION=rsh bandit "lp -d ps"  
SIRIUS=rsh bandit "lp -d ps -T pcl5"  
LOCAL=lp -d ps
```

In this example, both printers are connected to the system *bandit*, so the print command is a remote shell command executed on *bandit*. ORION is a PostScript printer, so the command **lp -d ps** is executed on *bandit* to print to ORION. SIRIUS, however, is a PCL5 printer, so the print command executed on *bandit* to print to SIRIUS is **lp -d ps -T pcl5**. There is also an entry for a printer connected to your local system.

Your printer port can be any name you choose except FILE:, which is the only reserved port name. It causes Designer to create a print file formatted specifically for the specified printer type.

- 7. Add/Replace.** The new port is now included in the list of current port definitions.
- 8. Select the printer from the Printer Device list box and the desired port in the Current Port Definitions list box.** If no description matches your printer, contact Actel for a printer description (PPD) file and install it in the <actel directory>/xprinter/ppds directory.
- 9. Click Add Selected and Dismiss.** This will take you back to the Printer Installation dialog box.
- 10. Click Dismiss.** The Printer Setup dialog box is displayed.

Not that you've specified available printers, you can make one of them the default printer.

To specify a default printer:

1. Click the **Properties** button from in the Printer dialog box.
2. Click the **Options** button in the Printer Setup dialog box.
<display option dialog box>
3. From the Printer Name drop-down list, select the desired printer and click **OK**.
4. Click **Save** on the Printer Setup dialog.

To set printer options:

- The specific options available vary between printers.
1. Click the **Properties** button in the Printer dialog box.
 2. Set all fields to the desired values. The following table describes all printer setup fields:

Table B-1. Printer Setup Options

Option	Description
Output Format	Specify whether to send output to a file or to a printer. If you choose Printer Specific, you can send output to any available printer. If the port is FILE:, Designer creates an output file specifically for the specified printer type. If you choose Generic (File Only), print output is sent to an Encapsulated PostScript or generic PCL file.
Printer	This field only appears if you select Output Format: Printer Specific. It specifies the name of the default printer to send print output to. Click the Options button to specify a different printer.
File Name	This field only appears if you select Output Format: Generic (File Only). Type the name of the print file you wish to create. To pipe print output to a command, type a ! character as the first character and then specify the command to pipe output to. For example, to pipe output to the lp command, enter the following: !lp -s ps .

Table B-1. Printer Setup Options (Continued)

Option	Description
EPSF PCL4 PCL5	This field only appears if you select Output Format: Generic (File Only). Click this button to display a list of output file types and select the desired type. Available types are EPSF (Encapsulated PostScript), PCL4, and PCL5.
Orientation	Specify portrait or landscape.
Scale	To increase the size of the output, specify a value greater than 1.00. To reduce the size, specify a value less than 1.00. For example, a value of 2.00 would double the size of the output; a value of 0.50 would reduce it by half.
Copies	Specify the number of copies to print.

3. **Click *Options* to set additional options, such as selecting a new printer or changing the page size.**
4. **Set all options to the desired values.** The following table describes all printer options:

Table B-2. Additional Printer Options

Option	Description
Printer Name	Use to change the Printer on the Setup dialog. Click the down arrow to display a list of available printers.
Resolution	Specify printer resolution. Values vary among different printers.
Page Size	Specify page size. Values vary among different printers.
Paper tray	Specify tray where paper is located. Values vary among different printers.

5. **Click *Save* to make your changes take effect and make them the new default values, or click *Apply* to make your changes take effect without changing the default values.**



Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Actel U.S. Toll-Free Line

Use the Actel toll-free line to contact Actel for sales information, technical support, requests for literature, Customer Service, investor information, and using the Action Facts service.

The Actel toll-free line is (888) 99-ACTEL.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call (408) 522-4480.

From Southeast and Southwest U.S.A., call (408) 522-4480.

From South Central U.S.A., call (408) 522-4434.

From Northwest U.S.A., call (408) 522-4434.

From Canada, call (408) 522-4480.

From Europe, call (408) 522-4252 or +44 (0) 1276 401500.

From Japan, call (408) 522-4743.

From the rest of the world, call (408) 522-4743.

Fax, from anywhere in the world (408) 522-8044.

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Guru Automated Technical Support

Guru is a web-based automated technical support system accessible through the Actel home page (<http://www.actel.com/guru/>). Guru provides answers to technical questions about Actel products. Many answers include diagrams, illustrations, and links to other resources on the Actel web site.

Web Site

Actel has a World Wide Web home page where you can browse a variety of technical and non-technical information. The URL is <http://www.actel.com>.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Electronic Mail

You can communicate your technical questions to our e-mail address and receive answers back by e-mail, fax, or phone. Also, if you have design problems, you can e-mail your design files to receive assistance. We constantly monitor the e-mail account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support e-mail address is **tech@actel.com**.

Telephone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

(408) 522-4460

(800) 262-1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Please see our list of [Worldwide Sales Offices](#).

Worldwide Sales Offices

Headquarters

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
Toll Free: 888.99.ACTEL
Tel: 408.739.1010
Fax: 408.739.1540

US Sales Offices

California

Bay Area
Tel: 408.328.2200
Fax: 408.328.2358

Irvine
Tel: 949.727.0470
Fax: 949.727.0476

Newbury Park
Tel: 805.375.5769
Fax: 805.375.5749

Colorado

Tel: 303.420.4335
Fax: 303.420.4336

Florida

Tel: 407.977.6846
Fax: 407.977.6847

Georgia

Tel: 770.277.4980
Fax: 770.277.5896

Illinois

Tel: 847.259.1501
Fax: 847.259.1575

Massachusetts

Tel: 978.244.3800
Fax: 978.244.3820

Minnesota

Tel: 651.917.9116
Fax: 651.917.9114

New Jersey

Tel: 609.517.0304

North Carolina

Tel: 919.654.4529
Fax: 919.674.0055

Pennsylvania

Tel: 215.830.1458
Fax: 215.706.0680

Texas

Tel: 972.235.8944
Fax: 972.235.9659

International Sales Offices

Canada

235 Stafford Rd. West, Suite 106
Nepean, Ontario K2H9C1, Canada
Tel: 613.726.7575
Fax: 613.726.8666

France

361 Avenue General de Gaulle
92147 Clamart Cedex
Tel: +33 (0)1.40.83.11.00
Fax: +33 (0)1.40.94.11.04

Germany

Lohweg 27,
D-85375 Neufahrn
Germany
Tel: +49.(0)81.659.584.0
Fax: +49.(0)81.659.584.10

Italy

Via dei Garbaldini 5
20019 Settimo Milanese
Milano, Italy
Tel: +39 (0)2.3809.3259
Fax: +39 (0)2.3809.3260

Japan

EXOS Ebisu Building 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150
Tel: +81 (0)3.3445.7671
Fax: +81 (0)3.3445.7668

Korea

30th floor, ASEM Tower,
159-1 Samsung-dong,
Kangnam-ku, Seoul, Korea
Tel: +82 (0)2.6001.3382
Fax: +82 (0)2.6001.3030

United Kingdom

Maxfli Court
Riverside Way
Camberley, Surrey
GU15 3YL
United Kingdom
Tel: +44 (0)1276.401450
Fax: +44 (0)1276.401490

Index

A

Actel
 Manuals xiii
 web site 156
 web-based technical support 156
ADB
 file association 84
ASICmaster
 constraint file
 syntax, conventions for 142
 syntax, summary 142
Assigning pins, see PinEdit
Assumptions xii
Auditing files 37
Auxiliary files 30
 importing 31
Axcelerator
 layout 50
 PDC files 33
 PinEdit 47

B

Back-Annotation 58
Bitstream 62

C

Changing design information in Designer 65
ChipEdit 47
Combine registers 46
Compile 25, 44
 new design 25
Compile options 45
 abort on PDC error 46
 combine registers 46
 fanout warning limit 46
 netlist pin properties overwrite 46

Contacting Actel
 customer service 155
 electronic mail 156
 telephone 157
 toll-free 155
 web-based technical support 156
Customer service 155

D

Design
 compiling 44
 designs created in previous versions 27
 new 25
Design information
 changing in Designer 65
Design session
 starting 24
Design setup 65
Designer
 back-annotation 58
 changing design information 65
 ChipEdit 47
 compile 25
 compiling a new design 25
 designs created in previous versions 27
 exiting 88
 exporting files 67
 flip flop report 78
 importing a design 25
 incremental placement 55
 layout 50
 layout failures 57
 new design 25
 pin report 77
 preferences 83
 reports 77

- selecting a package 66
- selecting die 66
- selecting speed grade 66
- SmartPower
- starting 24
- status report 72
- Timer 48
- timing report 72
- tools 47
- Device
 - selection 66
- Device selection wizard 39
 - changing device, package, and speed grade 66
- Document Assumptions xii
- Document Organization xi
- DT Layout, see Timing driven layout 51

E

- Effort level 57
- Electronic mail 156, 157
- exiting Designer 88
- Exporting
 - files 67
 - PDC files 69
 - STAMP 69
- Extended run 56

F

- Failures
 - layout 57
- Fanout
 - adjusting warning level 46
- File
 - association 84
- Files
 - auditing 37

- auxiliary 30
- bitstream 62
- exporting 67
- fuse 61
- PDC 33
- SDC 35
- source 27
- STAMP 69
- Flip flop report 78
- Fuse 61

G

- Generating
 - bitstream files 62
 - fuse files 61
 - programming files 60
 - reports 72
 - reports in Designer 77

I

- I/O placement, see ChipEdit
- Importing
 - auxiliary files 31
 - PDC 34
- Importing SDC files 35
- Incremental placement 55
- Internet 84
 - proxy 84

L

- Layout 50
 - antifuse 50
 - effort level 57
 - extended run 56
 - failures 57
 - incremental placement 54, 55

- router options 56
- standard 53
- timing driven 54
- timing driven failures 57
- timing weight 57

N

- Netlist
 - importing into Designer 25
 - viewing, see Netlist Viewer
- Netlist pin properties 46
- Netlist Viewer
 - Designer
 - Netlist Viewer 48
- New design 25

O

- Online Help xiii

P

- PDC
 - abort on import error 46
 - exporting 69
 - tcl commands 100
- PDC (Physical Design Constraint) 33
- PDF reader 85
- Physical Design Constraint (PDC) 33
- Pin
 - assigning 47
 - printing a list 77
- pin 77
- PinEdit
 - description 47
- Placement
 - incremental 54
- Power

- report 80
- Power analysis, see SmartPower
- Preferences
 - designer 83
 - internet 84
- Printing
 - pin list 77
 - timing information 72
- ProASIC Layout Viewer 48
- Product support 155–158
 - customer service 155
 - electronic mail 156, 157
 - oll-free line 155
 - technical support 156
 - web site 156
- Programming files 60
 - bitstream 62
 - fuse 61
- Proxy 84

R

- Related Manuals xiii
- Report
 - timing 72
- Reports 77
 - Designer 77
 - flip flop 78
 - generating 72
 - power 80
 - status 72
 - timing 72
 - timing violations 76
- reports 72

S

- Scripting, see also Tcl

SDC (Synopsys Design Constraints) 35
Selecting
 die 66
 package 66
 speed grade 66
Setup design
 changing design name and family 65
SmartPower 48
Source files 27
 importing 28
Stamp 69
Standard layout 51, 53
Starting 24
 new design 25
Static timing analysis, see Timer
Status 72
Status report 72
Synopsys Design Constraints (SDC) 35
syntax conventions
 comments 142
 notation 142
 KEYWORD 142
 variables 143
 FILEIDENTIFIER 143
 IDENTIFIER 143
 POSFLOAT 143
 POSNUMBER 143
 QPATH 143
syntax, for constraint file 142

T

Tcl
 command origins 93
 command substitution 91
 commands
 compile 95

 export 95
 get_defvar 96
 import 96
 import_source 97
 layout 97
 new_design 98
 open_design 98
 report 98
 save_design 99
 set_defvar 99
 set_design 99
 set_device 99
control structures 92
description 89
Designer commands 95
other features 94
overview 89
PDC 100
PDC commands
 reset_io_command 100
 reset_iobank 100
 reset_net_criticality 100
 set_io 100
 set_iobank 101
 set_location 102
quotes and braces 91
syntax 89
variables 90
Timer 48
Timing
 reports 72
 violations report 76
Timing driven layout 51
Timing report 72
Toll-free line 155

U

UNIX

 PDF reader 85

 web browser 85

User Tools 47

V

Violations

 timing report 76

W

Web browser 85

Web-based technical support 156

Weight

 timing 57

