Designer v5.0
User's Guide

Actel® Corporation, Mountain View CA 94043-4655 USA© 2002 Actel Corporation. All rights reserved.

Part Number: 5029122-5

Release: August 2003

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.
Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.
This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

Trademarks

Actel and the Actel logotype, 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

Welcome to Designer

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

The basic steps to implement your design include:

- Starting a new design
- Importing your source and auxiliary file(s)
- Compiling your design
- Running Layout to place-and-route your design
- Back-Annotating your design
- Generating your programming file (Fuse or Bitstream)
- Programming your device

Designer also includes the following User Tools that can be used to optimize your design:

- PinEditor package level floorplanner and I/O attribute editor
- ChipEditor chip level module placer
- NetlistViewer design schematic viewer
- SmartPower power analysis tool
- Timer static timing analysis and constraints editor
- Timing driven place and route
- ACTgen macro generator
- Silicon Explorer II In-system probing software*
- Silicon Sculptor II device programming software*

These tools give expert users maximum flexibility to drive the place-and-route tools to achieve the timing required.

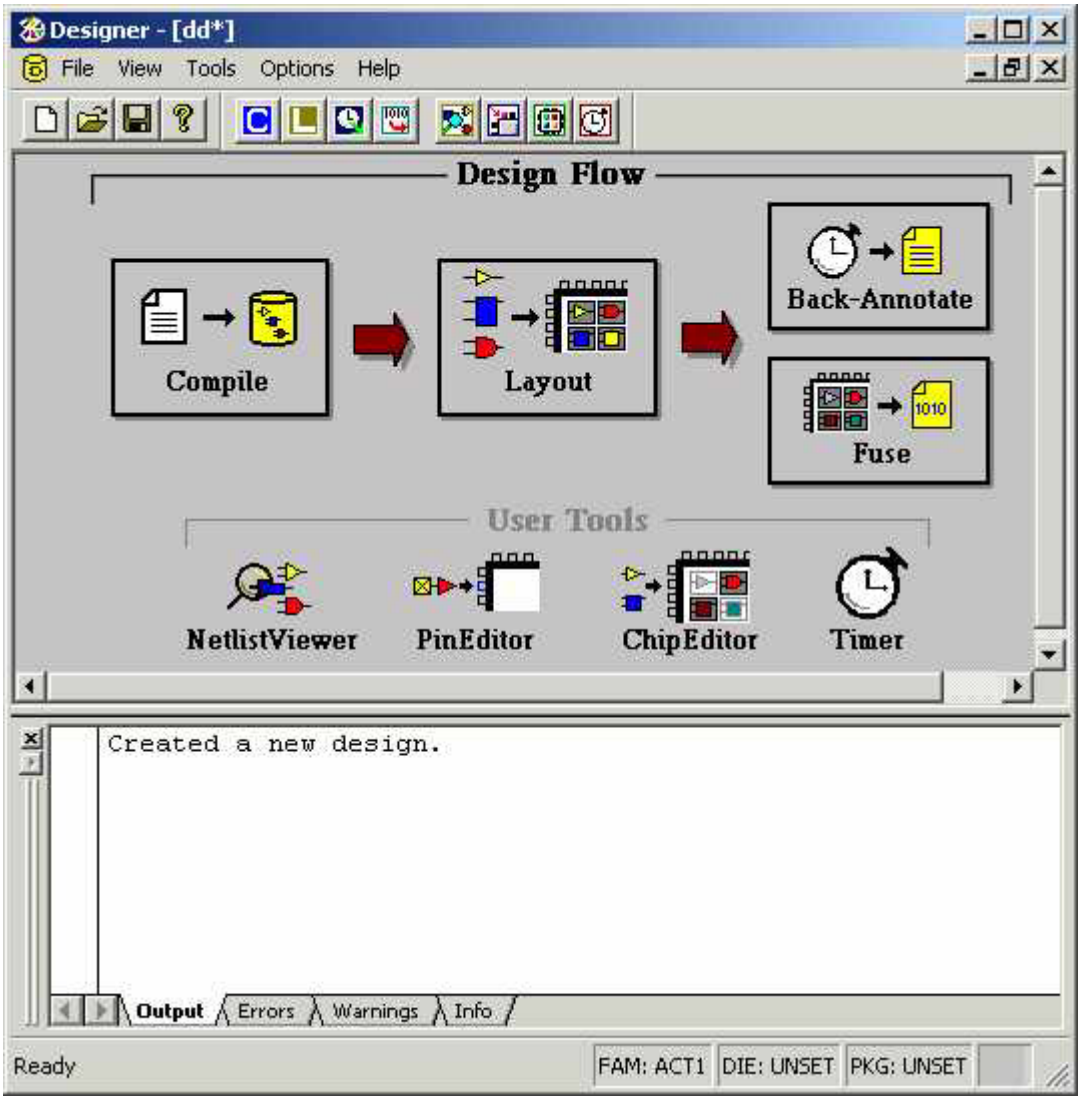
Getting Started

Starting a new design

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

To start a new design:

1. Click **Start New Design** in the Designer main window, or in the **File** menu, click **New**. This displays the Setup Design dialog box.
2. Setup Design:
   - Enter a **Design Name**. The design name is used in reports and as the default name when saving or exporting files.
   - Select an **Actel Family** from the drop down menu list.
   - Specify a **working directory**. Click Browse to locate a directory.
3. Click **OK**. The Designer custom design flow window appears. All tools and commands are activated.



Designer: New Design

Opening an existing design

To open an existing design:

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

Opening designs created in previous versions

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

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

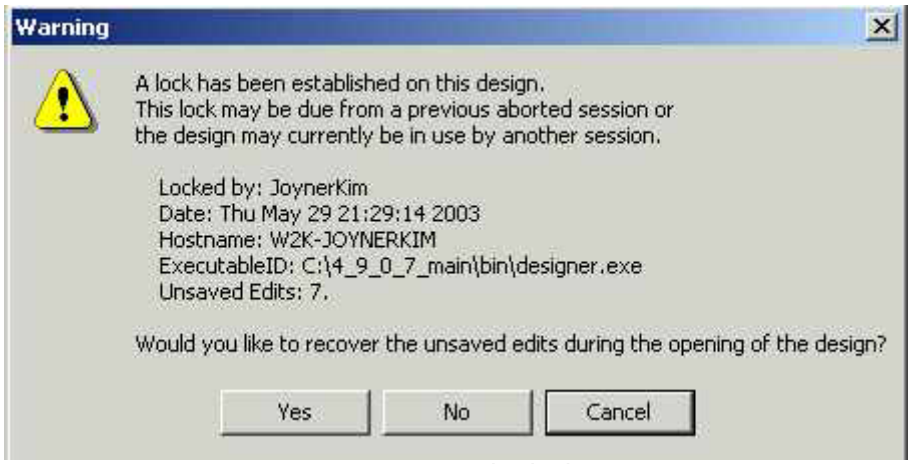
All existing die, package, pin assignments, and place-and-route information is read and maintained. Designs created in previous versions of software may need library conversions when loaded into the Designer environment. If your design requires this conversion, Designer prompts you to allow the software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

Opening locked files

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

Warning

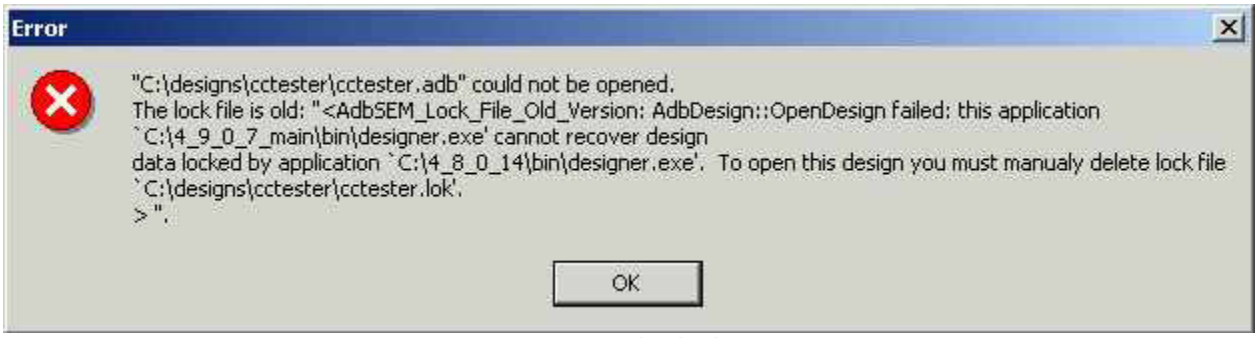
Designer warns you when opening a design that was not closed properly or may be open somewhere else. You can choose to recover the unsaved edits.



Warning: Locked File

Error

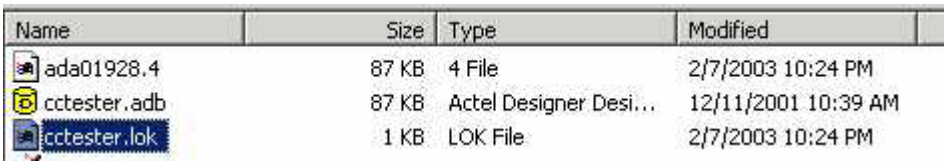
When opening a design, an error might notify you that the file can't be opened because the lock file is old. You can't recover any unsaved edits.



Error: Locked File

To open a design with an old lock file:

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



Locked File

Starting other applications from Designer (PC only)

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

To add an application to the Tools menu:

1. From the **Tools** menu, click **Customize**.
2. Enter the application name in the Menu Text area. This text will appear in the Tools command menu.

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

To remove an application from the Tools menu:

1. From the **Tools** menu, click **Customize**.
2. Select the application to remove and click **Remove**.
3. When you are finished removing applications, click **OK**.

To order applications in the Tools menu:

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

License details

To display information about your license:

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

Preferences

Directory preferences

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

To specify your directory preferences:

1. From the **File** menu, click **Preferences**.
2. Click the **Directory** tab.
3. Specify your Startup directory.
4. Select your working directory options:
5. **To design file's directory when opening design**: Select to automatically change directories when opening a design.
6. **To design file's directory when saving design**: Select to automatically change directories when saving a design.
7. **To script file's directory when executing script**: Select to automatically change directories when executing a script.
8. **Add design name to working directory when creating design**: Select to enable a design name folder to be automatically created in the working directory when creating a new design.
5. Click **OK**.

Updates

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

To set your automatic software update preferences:

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

- **Automatically check for updates at startup:** Select to be notified of updates when you start Designer.

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

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

Note:

- This feature requires an internet connection.

Proxy

A Proxy improves access to the Actel server.

To enable the proxy:

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

File association

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

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

To associated .adb files with the Designer application:

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

Setting your Log Window preferences

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

To change colors in the log window:

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

The default color settings for the log window are:

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

PDF Reader (UNIX Only)

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

Device Selection

Use the Device Selection Wizard to select your die, package, speed, voltage, and operating conditions.

Device Selection Wizard

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

The wizard steps include:

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

Setting die, package, speed, and voltage

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

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

Device variations

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

To select reserve pins and default I/O standard:

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

Reserve Pins are not selectable for the Axcelerator, ProASIC, and ProASIC Plus families.

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

Setting Operating Conditions

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

Junction Temperature

Select a junction temperature. Supported ranges include:

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

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

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

Voltage

Select a voltage:

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

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

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

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

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

Radiation Derating

Conservative post radiation performance estimates are available for some radiation tolerant devices based upon the number of KRads the device is expected to be subjected to. Radiation effects vary by device lot and may not be completely representative of the lot you are using. Post

radiation timing numbers are only meant to be a guide and are not a guarantee of performance. Customers must consult the specific radiation performance report for the specific lot used. Post radiation exposure estimates currently only affect timing numbers. The SmartPower power analysis tool is not affected by changing the radiation exposure value.

Changing design name and family

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

To change the design name or family:

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

Changing device information

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

To change design information for existing designs:

1. In the **Tools** menu, click **Device Selection**. The Device Selection Wizard appears.
2. Select Die, Package, and Speed Grade and click **Next**. (You must select die and package to continue.)
3. Select Device Variations and click **Next**.
4. Select Operating Conditions and click **Finish**.

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

Changing Device, Package, and Speed Grade

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

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

Compatible Die Change

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

Changing Die Revisions

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

Device Change Only

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

Repackager Function (Non-Axcelerator families only)

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

Importing Files

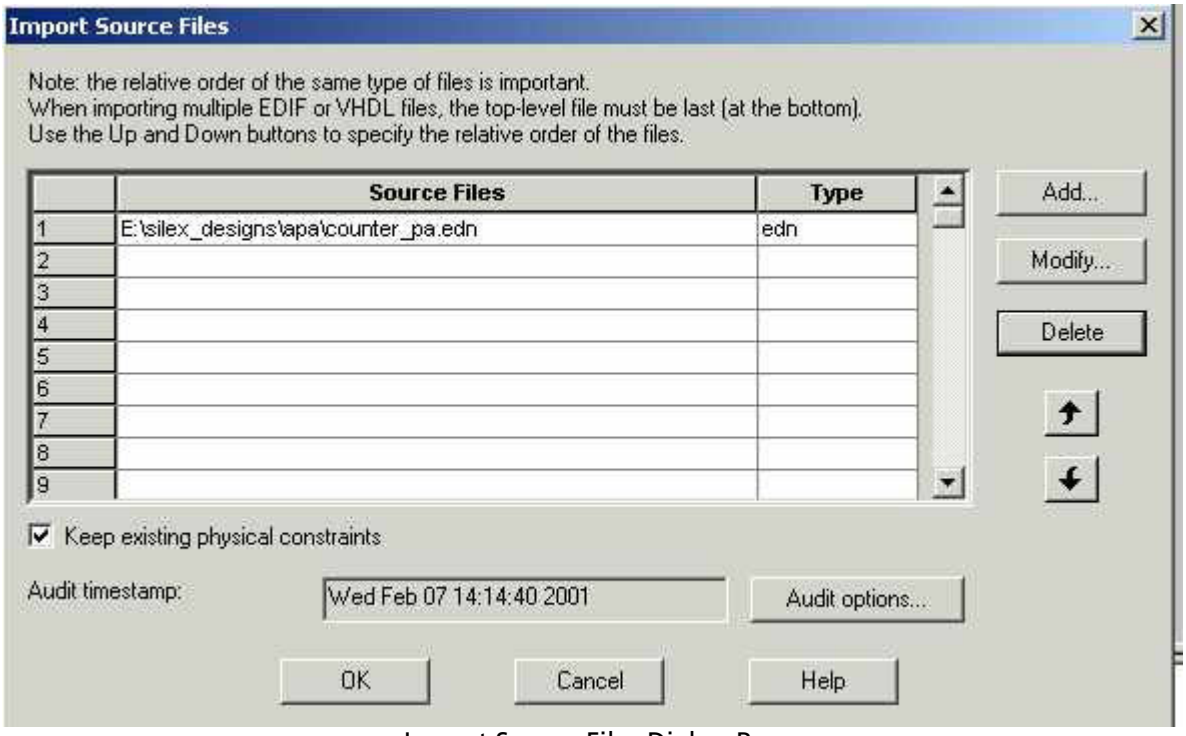
Source files include your netlist and constraint files.

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

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

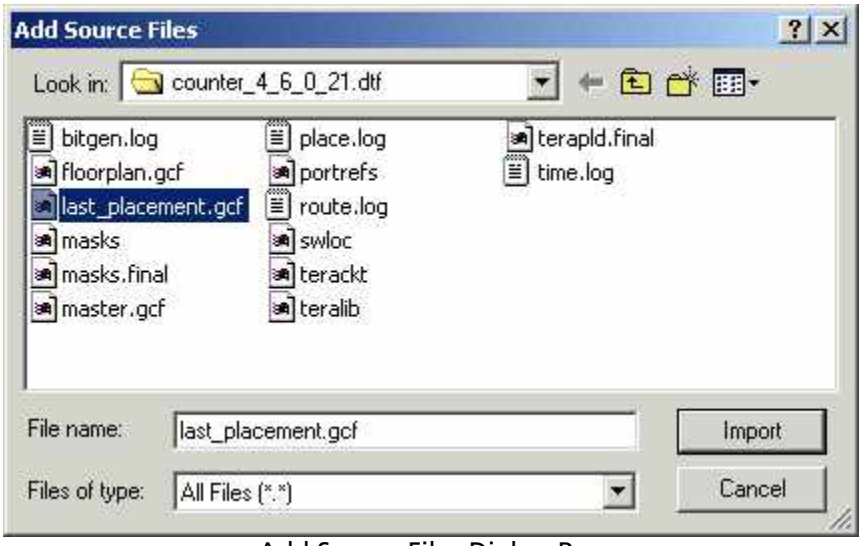
To import a source file:

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



Import Source Files Dialog Box

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



Add Source Files Dialog Box

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

Note:

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

Auditing files

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

To change your audit settings:

1. From the **File** menu, click **Audit Settings**. The Audit Settings dialog box appears. Audit Timestamp reflects the last time and day that the import source or audit update was successfully done.
2. Select the audit check box next to the file to enable auditing.
3. Click **Change Location** to move the file to another directory.
4. Click **Reset to Current Date Time** to associate the file with the current day and time.

Importing Constraint (Auxiliary) Files

The following constraint file types can be imported into Designer:

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

To import an auxiliary file:

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

Note:

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

Importing PDC files (Axcelerator family only)

The Physical Design Constraint (PDC) file can specify:

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

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

To import a PDC file:

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

Note:

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

Importing Synopsys Design Constraint files

SDC is a Tcl-based format-constraining file. The commands of an SDC file follow the Tcl syntax rules. Designer accepts an SDC constraint file generated by a third-party tool.

To import an SDC file:

1. From the **File** menu, click **Import Auxiliary Files**. The Import Auxiliary Files dialog box is displayed.
2. Click **Add**. The Add Auxiliary Files dialog box appears.
3. Select your SDC file. Filter for SDC files by selecting SDC Files in the Files of Type drop-down list box.
4. Click **Import**. The SDC file is added to the Import Auxiliary Files dialog box.
5. Click **OK**. The SDC file is imported into your design. Any errors appear in the Log Window.

When Compile and Layout complete and Timer starts, the constraints from the SDC file are incorporated in the timing of the design and are reflected in Timer.

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

Compile

Compiling your design

After you import your netlist files and select your device, you must compile your design. Compile contains a variety of functions that perform legality checking and basic netlist optimization. Compile checks for netlist errors (bad connections and fan-out problems), removes unused logic (gobbling), and combines functions to reduce logic count and improve performance. Compile also verifies that the design fits into the selected device.

There are three ways to select the compile command:

- In the Tools menu, click **Compile**.
- Click the **Compile** button in the Design Flow.
- Click the **Compile** icon in the toolbar.

If you have not already done so, Designer's Device Selection Wizard prompts you to set the device and package.

During compile, the message window in the Main window displays information about your design, including warnings and errors. Designer issues warnings when your design violates recommended Actel design rules. Actel recommends that you address all warnings, if possible, by modifying your design before you continue.

If the design fails to compile due to errors in your input files (netlist, constraints, etc.), you must modify the design to remove the errors. You must then re-import and re-compile the files.

After you compile the design, you can run Layout to place-and-route the design or use the User Tools (PinEditor, ChipEditor, ChipPlanner, Timer, SmartPower, or NetlistViewer) to perform additional optimization prior to place-and-route.

Compile Options

The compile options are specific to each family. Compile options are not available for the ProASIC and ProASIC PLUS families.

To set compile options:

1. From the **Options** menu, click **Compile**. The compile option dialog box opens. Options available are family specific.
2. Select your options and click **OK**.

Netlist Pin Properties Overwrite Existing Properties

During the Compile process, Designer checks the netlist properties. If the netlist file specifies a pin assignment for a pin that was also assigned in PinEditor session, there is a conflict. How this conflict is resolved is determined by your selection in this box.

If this option is off, or unchecked, then Designer uses the assignment made in PinEditor and the assignment in the netlist file for the conflicting pin is ignored.
If this option is on, or checked, then Designer uses the assignment in the netlist file for that pin and the PinEditor assignment is ignored. If you edit pin assignments in PinEditor, this option is automatically set to "off."

Combine Registers into I/Os (Axcelerator Family)

The Axcelerator family includes an optional register on the input path, an optional register on the output path, and an optional register on the 3-state control pin.
Select the option Combine Registers into I/Os where possible to take advantage of these registers.

Abort on PDC Error (Axcelerator Family)

Setting Abort on PDC Error aborts the PDC import when an error is encountered. When this box is checked, the PDC file is either imported fully or the design is left untouched.

Fanout Messages (ACT1, ACT2, ACT3, DX, MX, SX, SX-A, eX)

Use the control slider in the Messages area to control the warning level. Use the control slider to specify the fanout limit that the Compile step checks against. Setting the control slider to '0'

informs the system to use the system defaults. Any non-zero value replaces the system default value for the fanout limit with the user-specified value. Typically, this value range is 1 to 24.

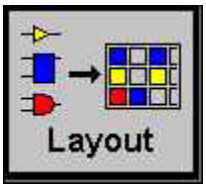
This does not adjust the fanout of the design and it has no effect on the netlist. This only adjusts the warning level, by controlling what level of fanout checking you want to be warned about during Compile. Changing this fanout limit option does not invalidate the Compile design state.

Layout

Use Layout to place and route your design.

To run Layout:

1. Click the **Layout** button in the Design Flow Window.



2. **Layout Options**. Select your Layout options and Click **OK**. Layout options are family specific.

Axcelerator Layout Options

When running Layout, use the Layout dialog box to set your Layout options.

Timing-driven

Select this option to run timing-driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. timing-driven Layout is more precise and typically results in higher performance.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if timing-driven Layout is required.

Run Place

Select this option to run the placer during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Place is checked by default. If your design has already

been placed, this box is not checked. You can also select the following incremental placement options.

- **Incrementally**: Select to use previous placement data as the initial placement for the next placement run.
- **Lock Existing Placement (fix):** Select to use and fix previous placement data for the next incremental placement run.

Effort Level

Use the Effort Level slider to increase the effort Layout uses to place and route your design. The range is 1 to 5 with a default of 3. A higher level of effort generally improves the quality of results, but runs longer.

Run Route

Select to run the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Route is checked. Run Route is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this box is checked.

Use Multiple Passes

Select to run layout multiple times with different placement seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

Note: To run Multiple Passes, you must check both Run Place and Run Route.

Flash Layout Options

When running layout, use the Layout dialog box to set your layout options.

Timing-driven

Select this option to run timing-driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. Timing-driven Layout is more precise and typically results in higher performance.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if timing-driven Layout is required.

Run Place

Select this option to run the placer during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Place is checked by default. If your design has already been placed but not routed, this box is not checked by default. You can also select the following incremental placement options.

- **Incrementally**: Select to use previous placement data as the initial placement for the next place run.
- **Lock Existing Placement (fix):** Select to preserve previous placement data during the next incremental placement run.

Run Route

Select to run the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Route is checked. Run Route is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this box is checked.

- **Incrementally**: Select to fully route a design when some nets failed to route during a previous run. You can also use it when the incoming netlist has undergone an E.C.O. (Engineering Change Order). Incremental routing should only be used if a low number of nets fail to route (less than 50 open nets or shorted segments). A high number of failures usually indicates a less than optimal placement (if using manual placement through macros, for example) or a design that is highly connected and does not fit in the device. If a high number of nets fail, relax constraints, remove tight placement constraints, or select a bigger device and rerun routing.

Use Multiple Passes

Select to run layout multiple times with different seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

eX, SX, SX-A Layout Options

When running layout, use the Layout dialog box to set your layout options.

Timing-Driven

Select to run Timing-Driven Layout. The primary goal of Timing-Driven layout is to meet timing constraints, while still producing high performance for the rest of the design. Timing-Driven Layout is more precise and typically results in higher performance. This option is available only when timing constraints have been defined.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if Timing-Driven Layout is required.

Place Incrementally

Select to use previous placement data as the initial placement for the next place run.

- **Lock Existing Placement:** Select to preserve previous placement data during the next incremental placement run.

Use Multiple Passes (eX and SX-A only)

Select to run layout multiple times with different seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

Advanced

Click the Advanced button to set Extended Run and Timing-Driven options.

eX, SX, and SX-A Advanced Layout Options

To set these advanced options during Layout, click the Advanced button in the Layout dialog box.

Extended Run

Select this to run a greater number of iterations during optimization within a single layout pass. An extended run layout can take up to 5 times as long as a normal layout.

Effort Level

This setting specifies the duration of the timing-driven phase of optimization during timing-driven Layout. Its value specifies the duration of this phase as a percentage of the default duration. This option is available only when timing constraints have been defined

The default value is 100 and the selectable range is within 25 - 500. Reducing the effort level also reduces the run time of timing-driven place-and-route (TDPR). With an effort level of 25, TDPR is almost four times faster. With fewer iterations, however, performance may suffer. Routability may or may not be affected. With an effort level of 200, TDPR is almost two times slower. This variable does not have much effect on timing.

Timing Weight

Setting this option to values within a recommended range of 10-150 changes the weight of the timing objective function, thus influencing the results of timing-driven place-and-route in favor of either routability or performance. This option is available only when timing constraints have been defined

The timing weight value specifies this weight as a percentage of the default weight (i.e. a value of 100 has no effect). If you use a value less than 100, more emphasis is placed on routability and less on performance. Such a setting would be appropriate for a design that fails to route with

TDPR. In case more emphasis on performance is desired, set this variable to a value higher than 100. In this case, routing failure is more likely. A very high timing value weight could also distort the optimization process and degrade performance. A value greater than 150 is not recommended.

Restore Defaults

Click **Restore Defaults** to run the factory default settings for advanced options.

ACT, MX, and DX Layout Options

Timing-driven

Select this option to run Timing-Driven Layout. The primary goal of timing-driven layout is to meet timing constraints, with a secondary goal of producing high performance for the rest of the design. timing-driven Layout is more precise and typically results in higher performance. This option is available only when timing constraints have been defined.

When not checked, standard layout runs. Standard layout maximizes the average performance for all paths. Each part of a design is treated equally for performance optimization. Standard layout uses net weighting (or criticality) to influence the results. Delay constraints that have been set for a design during place-and-route are not considered, however a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if Timing-Drive Layout is required.

Place Incrementally

Select to use previous placement data as the initial placement for the next place run.

- **Lock Existing Placement:**Select to preserve previous placement data during the next incremental placement run.

Advanced

Click Advanced to set Extended Run options.

ACT, MX, and DX Advanced Layout Options

To set these advanced options during Layout, click the Advanced button in the Layout dialog box.

Extended Run

Select this to run a greater number of iterations during optimization. An extended run layout can take up to 5 times as long as a normal layout

Restore Default

Click **Restore Defaults** to run the factory default settings for advanced options.

Incremental Placement

In either standard or timing-driven mode, use incremental placement to preserve the timing of a design after a successful place and route, even if you change part of the netlist. Incremental placement has no effect the first time you run layout. During design iteration, incremental placement attempts to preserve the placement information for any unchanged macros in a modified netlist.

As a result, the timing relationships for unchanged macros approximate their initial values, decreasing the execution time to perform Layout. By forcing Designer to retain the placement information for a portion of the design, some flexibility for optimal design layout may be lost. Therefore, do not use incremental placement to place your design in pieces. You should only use it if you have successfully run Layout and you have minor changes to your design.

Incremental placement requires prior completion of place. Do not use incremental placement if the previous Layout failed to meet performance goals.

Locking Existing Placement (Fix)

When this option is selected in the Layout dialog box, all unchanged macros are treated as fixed placements during an incremental placement. This is the strongest level of control, but it may be

too restrictive for the new placement to successfully complete. The default ON setting treats unchanged macro locations as placement hints, but alters their locations as needed to successfully complete placement. Refer to ChipEditor for details on fixing macros.

Flash Placement Constraint File (GCF)

For Flash designs, the Designer software always produces a placement constraints file in the design directory called: <design>.dtf/Last_placement.gcf. This file contains all the information about the latest placement. Blocks with fixed placement constraints generate fixed placement constraints, while the others generate initial placement constraints. The existing constraint files can be edited to remove any prior placement constraints. The GCF command

```
read "last_placment.gcf";
```

can be added to an existing constraint file to indicate that the latest placement is to be used as the initial placement.

Move or copy "last_placment.gcf" to use it as an input constraint file. Otherwise, it is overwritten by any subsequent placement if it is left in its original location.

Multiple Pass Layout

Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of Layout results. This is done by running individual place and route multiple times with varying placement seeds and measuring the best results with a specified criteria.

Note:

- Before running Multiple Pass Layout, you need to save your design.
- Multiple Pass Layout is supported in the following families: Axcelerator, ProASIC, ProASICPLUS, SX-A, and eX.
- Multiple Pass Layout saves your design file with the pass that has the best layout results. A corresponding timing report file for the best result, named design-name_timing.rpt is also saved to disk. If you want to preserve your existing design state, you should save your

design file with a different name before proceeding. To do this, from the File menu, click Save As.

- A timing report for each pass will be written out to the working directory to assist you in later analysis. The report files will be named design-name_timing.rpt.pass-number. Look at the design-name _iteration_summary.rpt for details of the saved files.

To configure your multiple pass options:

1. When running Layout, se;ect **Use Multiple Passes** in the Layout Options dialog box.
2. Click **Configure**. The Multi-Pass Configuration dialog box appears.
3. Set the options and click **OK**.

   Maximum Number of Passes: Set the number of passes (iterations) using the slider. 3 is the minimum and 25 is the maximum. The recommended number of passes is 5.

   Measurement: Select the measurement criteria you want Layout to meet. If Slowest Clock or Specific Clock is selected as your criterion, then the Layout runs all passes. If Timing Violations is selected as your criterion, Layout stops once the timing constraints are met. If the constraints are not met, then all of the Layout passes run.

Slowest Clock	Select to use the slowest clock in the design in a given pass as the performance reference for the layout pass.
Specific Clock	Select to use a specific clock as the performance reference for all Layout passes.
Timing Violations	Select to use the pass that best meets the slack or timing-violations constraints. NOTE: You must enter your own timing constraints through the Timer or SDC.The 'best' case is calculated by determining the total negative slack for all constraints.

Save Results from All Passes: Select to save the design file (.adb) for each pass. By default, only the best result is saved to your design. With this option, for every pass, the individual .adb is stored as filename_pass-number.adb in the name. The 'best' pass design will still also be written back to the original .adb filename. Saving all results does take more disk space, but allows you to later analyze the result of each pass in more detail. Look at the design-name _iteration_summary.rpt for details of the saved files.

Back-Annotation

The back-annotation functions are used to extract timing delays from your post layout data. These extracted delays are put into a file to be used by your CAE package's timing simulator. If you wish to perform pre-layout back-annotation, select Export and Timing Files from the File menu.

The Back-Annotation program creates the files necessary for back-annotation to the CAE file output type that you chose. Refer to Actel Interface Guides or the documentation included with your simulation tool for information about selecting the correct CAE output format and using the back-annotation files.

To back-annotate your design:

1. From the **Tools** menu, click **Back-Annotate**, or click the Back-Annotate button in the Design Flow window.
2. Make your selections in the Back-Annotated dialog box and click **OK**.

   Extracted Files Directory: The file directory is your default working directory. If you wish to save the file elsewhere, click Browse and specify a different directory.

   Extracted File Names: This name is used as the base-name of all files written out for back-annotation. Do not use directory names or file extensions in this field. The file extensions will be assigned based on your selection of which file formats to export. The default value of this field is <design>_ba.

   Output Formats: Select SDF or STF (not supported for SX-A, eX, Axcelerator, ProASIC, and ProASIC PLUS.

   Simulator Language: Select either Verilog or VHDL93.

Export Additional Files: Check Netlist or Pin to export these files at the same time.

Exporting files

Designer supports the exporting of the following file types:

Files	File Extension	Family
Actel Flattened Netlist	.afl	All
Actel Internal Netlist	.adl	All
Standard Delay Format	.sdf	All
Standard Timing File	.stf	ACT1, ACT2, ACT3, MX, XL, DX, SX
STAMP	.mod, .data	SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Tcl Script File	.tcl	All
Verilog Netlist	.v	All
VHDL Netlist	.vhd	All
EDIF Netlist File	.edn	All
Log File	.log	All
STAPL	.stp	ProASIC, ProASIC PLUS
Bitstream	.bit	ProASIC, ProASIC PLUS
Data I/O Programming File (Legacy)	.dio	ACT1, ACT2, ACT3, XL, DX
Programming File (Legacy)	.fus	ACT1, ACT2, ACT3, MX, XL, DX
Actel Programming File	.afm	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Routing Segmentation File	.seg	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Silicon Explorer Probe File	.prb	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Placement Location File	.loc	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
ProASIC Constraints File	.GCF	ProASIC, ProASIC PLUS
Combiner Info	.cob	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Boundary Scan File	.bsd	DX, 42MX, SX, SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC PLUS
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC PLUS
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Designer does not support VHDL 87 in export.

To export a file:

1. In the **File** menu, click an export option from the **Export** sub-menu. Select the type of file you wish to export.
2. Specify file name and file type and click **OK**.

Generating Reports

Select from Report Type & Options on the dialog box when invoking a report.

Status Report provides information about Designer, Device Data, and variable settings for the design.

Timer Report displays summarized timing delays for paths. When Timing Report is selected and "OK" clicked, an additional dialog box prompts the user to select Timing Report Options.

Timing Violations Report summarizes timing violations.

Pin Report can be sorted by Name or Number. Select Pin Report, then click "OK" to set the List order.

FlipFlop Report can be Summary or Extended. Both reports include the Flip-Flop type, sequential (Seq) or combinatorial (CC), the Library name, and the Total number of Seq and CC Flip-Flops in the design. The Summary Report also includes the Number of instances of each unique type. The Extended Report provides the Macro name. All Reports are output to an editable window for viewing, modification, saving, and printing.

Power Reports allow you to quickly determine if any consumptions problems exist in your design.

Status Reports

The status report enables you to create a report containing device and design information, such as die, package, percentage of the logic and I/O modules used, etc.

To generate a status report:

1. In the Tools menu, click Reports.

2. Choose Status from the drop-down list in the Report Type dialog box. The status report opens in a separate window. You can save or print the report.

Timing reports

The timing report enables you to quickly determine if any timing problems exist in your design. The timing report lists the following information about your design:

- maximum delay from input I/O to output I/O
- maximum delay from input I/O to internal registers
- maximum delay from internal registers to output I/O
- maximum delays for each clock network
- maximum delays for interactions between clock networks

To generate a timing report:

1. In the **Tools** menu, click **Reports**.
2. Choose **Timing** from the Report Type drop-down list. This displays the Timing Report dialog box.
3. Specify the Slack Threshold. If you select “Slack” as the sort method, you can limit the number of delays displayed based upon a slack threshold. For example, if you only want to see delays that have a slack less than 5ns, enter 5 in the Slack Threshold box.
4. Setup-hold Timing Check. Selection of this box enables you to configure the timing report to calculate external setup and hold information for device inputs in addition to the standard information.
5. Expand Failed Paths. If a path does not meet your timing specifications, and you would like to see the incremental delay of each macro within that path, select the Expand Failed Paths box.
6. Options. Clicking Options brings up the Timing Preferences dialog box, where you can set additional display and report options.

Sort by Actual Delay

The actual delay is the path delay between two points in your design. This is the only way to sort your data if you do not have any timing constraints entered (for information on setting timing constraints, see the Timer User’s Guide). If you have entered timing

constraints, the actual delay report will automatically display the slack - even if you don't ask for it - but the data will always be listed from longest to shortest actual delay.
Actual delay measurements may be calculated before or after layout (that is, pre-layout or post-layout).

Sort by Slack Delay

Slack delay is the delay difference between a timing constraint entered in Timer and the actual delay of a path. For example, if a signal takes 20 ns to get from point A to point B, and you entered a timing constraint of 15 ns, the Timing Report would list -5 ns slack for that path. Thus, if the slack negative, then the actual delay did not meet the desired timing by the absolute value of the slack (in ns). Conversely, if the slack value is positive, then the timing constraint was met, with the slack value (in ns) to spare. In a slack report, the data will be sorted (by default) from longest to shortest slack.
When displaying slack, all the paths without timing constraints are filtered from the reported data. This allows you to quickly determine how well your design meets your timing requirements. This is especially useful for viewing critical delays like register-to-register, clock-to-out, and input-to-register.

Path Selection

Normally, only the longest path between any of the starting points (terminals) and each ending terminal is displayed. If you would like to see the timing of all paths between any of the starting terminals and any of the ending terminals, select Paths Between Any Pair in the Path Selection box.

Break Path at Register

The default timing paths break at all clock, gate, clear, and preset pins. If you would like to generate a timing report that passes through these pins, unselect the appropriate pins in the Break Path at Register options.

7. Click **OK**. This displays a timing report based upon your timing and display preferences. The format and content of the report is determined by the family

Pin reports

The pin report allows you to create a text list of the I/O signal locations on a device. You can generate a pin report sorted by I/O signal names or by package number.

To generate a pin report:

1. In the **Tools** menu, click **Reports**.
2. Choose **Pin** from the drop-down list in the Report Type dialog box. This displays the Pin Report dialog box.
3. Specify the type of report to generate. Select Number or Name from the List By pull-down menu, then click **OK**. This displays a pin report.

Flip-flop reports

The flip-flop report enables you to create a report that lists the number and type of flip-flops (sequential or CC, which are flip-flops made of 2 combinatorial macros) used in a design.

There are two types of reports you can generate, Summary or Extended:

A Summary report displays whether the flip-flop is a sequential, I/O sequential, or CC flip-flop, the macro implementation of the flip-flop, and the number of times the implementation of the flip-flop is used in the design.

An Extended report individually lists the names of the macros in the design.

To generate a flip-flop report:

1. In the **Tools** menu, click **Reports**. This displays the Reports dialog box.
2. Select Flip-Flop from the drop-down menu. The Flip-Flop Report dialog box appears.
3. Specify the type of report to generate. Select Summary or Extended from the Type pull-down menu, then click **OK**. This displays the report in a separate window.

Power reports

The power report enables you to quickly determine if any power consumption problems exist in your design. The power report lists the following information:

- Global device information and SmartPower Preferences selection information
- Design level static power summary
- Dynamic power summary
- Hierarchical detailed power report (including gates, blocks, and nets), with a block by block, gate by gate, and net by net power summary SmartPower results

To create a power report:

1. In the **Tools** menu, click **Reports**. This displays the Reports dialog box.
2. Choose **Power** in the Report list and click **OK**. The Power Report dialog appears.
3. Choose from the following options:

   • Static Power: Returns static power information

   • Dynamic Power: Returns dynamic power information

   • Report Style: Specifies report style

   For additional Power Report Options, click the Options to open the Power Preferences dialog box, as shown in Figure 2-45.

   Power Preferences Dialog Box

   Select analysis preferences:

   • Units: Sets units preferences for power and frequency

   • Operating Conditions: Sets preferences for operating conditions

   • Block Expansion Control: Filters reported power values returned in the report. This box does not control which values are included, rather it specifies which blocks are detailed/expanded. You may specify which blocks are expanded using a minimum power value, a minimum power ratio (with regards to the total power of the design) and a maximum hierarchical depth; a filtered value is not include in displayed lists, but still counted for upper hierarchical levels.
4. Once you are satisfied with your selections, click **OK** in the Preferences dialog box and then click **OK** in the Power Report dialog box. SmartPower displays the report in a separate window.

Timing Violations Reports

For families that use the pin-to-pin timing model, the Violations report enables you to obtain constraint results sorted by slack. You can now view Max Delay violations as well as Min Delay violations in the report.

To generate a timing violations report:

1. From **Tools** menu, click **Reports**.
2. In the Report Types dialog box, select **Timing Violations.**
3. Click **OK**.

Saving and Exiting

Saving your design

Once you have imported a netlist and compiled a design, you can save the design as an ADB file.

To save your design as an ADB file:

1. In the **File** menu, click **Save** or click the save icon in the toolbar.
2. Enter the File name and click **Save**. The default file name is the name you previously entered in the setup dialog box. The default format is adb. Make sure your save in the ".adb" format.

Once you have saved your compiled design as an ADB file, during any future Designer sessions, you can open the ADB file, skipping the compile step, and perform optimization on the design, including updating netlist and auxiliary file information.

Exiting Designer

To end a Designer session, from the **File** menu, click **Exit**.

If the information has not been saved to disk, you are asked if your want to save the design before exiting. If you choose YES, the "<design_name>.adb" file is updated with information entered the current session. If you choose NO, the information is not saved and the "<design_name>.adb" file remains unchanged.

Tcl Scripting

Tcl Documentation Conventions

The Actel command syntax conventions are as follows.

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

Tcl Commands

Designer supports the following Tcl scripting commands.

backannotate

The `backannotate` command is equivalent to executing the Back-Annotate command within the Tools menu. You can export an SDF file, after layout, along with the corresponding netlist in the VHDL or Verilog format. These files are useful in backannotated timing simulation.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
Backannote ?Option Option …?

-name file_name
-format format_type
-language
-simlang
-dir dir
-netlist
-pin
```

Arguments

`?-name file_name?`

Use a valid file name with this option. You can attach the file extension .sdf to the File_Name, otherwise the tool will append .sdf for you.

`?-format format_type?`

Only SDF format is available for back annotation

`?-language language?`

The supported Languages are options are

VHDL93 – For VHDL-93 style naming in SDF

VERILOG – For Verilog style naming in SDF

`?-dir directory_name?`

Specify the directory in which all the files will be extracted.

`?-netlist?`

Forces a netlist to be written. The netlist will be either in Verilog or VHDL depending on the

`-language option.`

The netlist file name will have the appropriate extension .v or .vhd appended to reflect the netlist format.

`?-pin?`

Designer exports the pin file with this option. The .pin file extension is appended to the design name to create the pin file.

Notes

We advise you to export both SDF and the corresponding VHDL/Verilog files. This will avoid name conflicts in the simulation tool.

Designer must have completed layout before this command can be invoked, otherwise the command will fail.

Exceptions

-pin is not supported for ProASIC and ProASICPLUS families.

Example

Example 1:

```
backannotate
Uses default arguments and exports SDF file for back
annotation
```

Example 2:

```
backannotate -dir \
 {..\my_design_dir} -name "fanouttest_ba.sdf" -format
"SDF" –language \ "VHDL93" –netlist
This example uses some of the options for VHDL
```

Example 3:

```
backannotate -dir \
 {..\design} -name "fanouttest_ba.sdf" -format "SDF" -
language "VERILOG" \
-netlist
```

```
This example uses some of the options for Verilog
```

Example 4:

```
If  { [catch { backannotate -name "fanouttest_ba" -format
"SDF" } ]} {
              Puts “Back annotation failed”
              # Handle Failure
} else {
              Puts “Back annotation successful”
              # Proceed with other operations
}

You can catch exceptions and respond based on the success
of backannotate operation
```

close_design

The close_design command closes the current design and brings Designer to a fresh state to work on a new design.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
close_design
```

Arguments

None.

Notes

This is equivalent to selecting the Close command in the File menu.

Exceptions

None.

Example

```
if  { [catch { close_design }] {
                Puts “Failed to close design”
                # Handle Failure
} else {
                puts “Design closed successfully”
                # Proceed with processing a new design
}
```

See Also

open_design, close_design, new_design

compile

The compile command performs design rule check on the input netlist. If the compile is successful, Designer reaches the compiled state. Compile also performs some optimizations on the design through logic combining and buffer tree modifications.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
compile ?argument argument …?
```

Arguments

`?-combine_register` *`value?`*`.`

Combines registers at the IO into IO-Registers. The value should be 1 for this optimization to take effect.

`?-nl_pins_overwrite?`

This option is used to overwrite the imported netlist with the changes made in PinEditor.

Notes

-combine_register option is available only for Axcelerator family.

-nl_pins_overwrite option is not available for Axcelerator, ProASIC and ProASIC[PLUS].

Exceptions

There are no compile options available for ProASIC and ProASICPLUS

Example

Example 1:

```
compile –combine_register
```

Example 2:

```
if  { [catch { compile –nl_pins_overwrite }] {
              Puts “Failed compile”
              # Handle Failure
} else {
              puts “Compile successful”
              # Proceed to Layout
}
```

export

The export command can be used to create a variety of files from Designer. The user through appropriate format and options can select these files for export. The basic types of files supported are listed in the following table.

Files	File Type Extension	Family
Actel Flattened Netlist	.afl	All
Actel Internal Netlist	.adl	All
Standard Delay Format	.sdf	All
Standard Timing File	.stf	ACT1, ACT2, ACT3, MX, XL, DX, SX
STAMP	.mod, .data	SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Tcl Script File	.tcl	All
Verilog Netlist	.v	All
VHDL Netlist	.vhd	All
EDIF Netlist File	.edn	All
Log File	.log	All
STAPL	.stp	ProASIC, ProASIC PLUS
Bitstream	.bit	ProASIC, ProASIC PLUS
Data I/O Programming File (Legacy)	.dio	ACT1, ACT2, ACT3, XL, DX
Programming File (Legacy)	.fus	ACT1, ACT2, ACT3, MX, XL, DX
Actel Programming File	.afm	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Routing Segmentation File	.seg	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A,

		eX
Silicon Explorer Probe File	.prb	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Placement Location File	.loc	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
ProASIC Constraints File	.GCF	ProASIC, ProASIC PLUS
Combiner Info	.cob	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX, Axcelerator
Boundary Scan File	.bsd	DX, 42MX, SX, SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Criticality	*.crt	ACT1, ACT2, ACT3, MX, XL, DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX
SDC	*.sdc	SX-A, eX, Axcelerator, ProASIC, ProASIC PLUS
Physical Design Constraint	*.pdc	Axcelerator
Value Change Dump	*.vcd	Axcelerator, ProASIC, ProASIC PLUS
Switching Activity Intermediate File/Format	*.saif	Axcelerator, ProASIC, ProASIC PLUS
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, XL, DX, SX, SX-A, eX

Supported Family and Format

Family: All

Format: TCL

Syntax

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

get_defvar

The get_defvar command provides access to the internal variables within Designer and returns it's value.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

get_defvar *variable*

Arguments

The *variable* is the Designer internal variable.

Notes

This command also prints the value of the Designer variable on the log window.

Exceptions

None.

Example

Example 1: Prints the design name on the log window.

```
 get_defvar “DESIGN”
set variableToGet "DESIGN"
set valueOfVariable [get_defvar $variableToGet]
puts "The value is $valueOfVariable"
```

See Also

set_defvar

get_design_filename

The get_design_filename command can be used to retrieve the full qualified path of the design file.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
get_design_filename
```

Arguments

None.

Notes

- The result will be an empty string if the design has not been saved to disk.
- This command is equivalent to the command "get_design_info DESIGN_PATH." This command predates get_design_info and is supported for backward-compatibility.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if arguments are passed.

Example

```
if { [ is_design_loaded ] } {
  set design_location [ get_design_filename ]
  if {$design_location != "" } {
    puts "Design is at $design_location."
  } else {
    puts "Design has not been saved to a file on disk."
  }
} else {
  puts "No design is loaded."
}
```

See Also

```
get_design_info
is_design_loaded
is_design_modified
is_design_state_complete
```

get_design_info

The get_design_info command can be used to retrieve some basic details of your design.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
get_design_info name | family | design_path |  cwdir | die |
package | speed | design_state
```

Arguments

The single argument must be one of the valid string values.

`name`

Design name.The result is set to the design name string.

`family`

Silicon family. The result is set to the family name.

`design_path`

Full qualified path of the design file. The result is set to the location of the .adb file. If a design has not been saved to disk, the result will be an empty string. This command replaces the command get_design_filename.

design_folder

Directory (folder) portion of the design_path.

design_file

Filename portion of the design_path.

`cwdir`

Current working directory.The result is set to the location of the current working directory

`die`

Die name.The result is set to the name of the selected die for the design. If no die is selected, this is an empty string.

`Package`

Package name.The result is set to the name of the selected package for the design. If no package is selected, this is an empty string.

`Speed`

Speed grade.The result is set to the speed grade for the design. If no speed grade is selected, this is an empty string.

Notes

The result value of the command will be a string value.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if more than one argument is passed.
- The command will return an error if the argument is not one of the valid values.

Example

The following example uses get_design_info to display the various values to the screen.

```
if { [ is_design_loaded ] } {
  puts "Design is loaded."
  set bDesignLoaded 1
} else {
  puts "No design is loaded."
  set bDesignLoaded 0
}
if { $bDesignLoaded != 0 } {
  set var [ get_design_info NAME ]
  puts "  DESIGN NAME:\t$var"
  set var [ get_design_info FAMILY ]
  puts "  FAMILY:\t$var"
  set var [ get_design_info DESIGN_PATH ]
  puts "  DESIGN PATH:\t$var"
  set var [ get_design_info DESIGN_FILE ]
  puts "  DESIGN FILE:\t$var"
  set var [ get_design_info DESIGN_FOLDER ]
  puts "  DESIGN FOLDER:\t$var"
```

```
    set var [ get_design_info CWDIR ]
    puts "  WORKING DIRECTORY:  $var"
    set var [ get_design_info DIE ]
    puts "  DIE:\t$var"
    set var [ get_design_info PACKAGE ]
    puts "  PACKAGE:\t'$var'"
    set var [ get_design_info SPEED ]
    puts "  SPEED GRADE:\t$var"
    if { [ is_design_modified ] } {
      puts "The design is modified."
    } else {
      puts "The design is unchanged"
    }
  }
  puts "get_design.tcl done"
```

See Also

```
  get_design_filename
  is_design_loaded
  is_design_modified
  is_design_state_complete
```

is_design_loaded

The is_design_loaded returns a Boolean value (0 for false, 1 for true) indicating if a design is loaded in the Designer software. True is returned if a design is currently loaded.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
is_design_loaded
```

Arguments

None

Notes

Some Tcl commands are valid only if a design is currently loaded in Designer. Use the 'is_design_loaded' command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if a design has been loaded.

```
set bDesignLoaded [ is_design_loaded ]
if { $bDesignLoaded == 0 } {
  puts "No design is loaded."
}
```

See Also

```
get_design_filename
get_design_info
is_design_modified
is_design_state_complete
```

is_design_modified

The is_design_loaded returns a Boolean value (0 for false, 1 for true) indicating if a design is loaded in the Designer software. True is returned if a design is currently loaded.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
is_design_loaded
```

Arguments

none

Notes

Some Tcl commands are valid only if a design is currently loaded in Designer. Use the is_design_loaded command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if a design has been loaded.

```
set bDesignLoaded [ is_design_loaded ]
if { $bDesignLoaded == 0 } {
  puts “No design is loaded.”
}
```

See Also

```
get_design_filename
get_design_info
is_design_modified
is_design_state_complete
```

is_design_state_complete

The is_design_state_complete command returns a Boolean value (0 for false, 1 for true) indicating if a specific design state is valid. True is returned if the specified design state is valid.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
is_design_state_complete SETUP_DESIGN | DEVICE_SELECTION |
NETLIST_IMPORT | COMPILE | LAYOUT | BACKANNOTATE |
PROGRAMMING_FILE
```

Arguments

The single argument must be one of the valid string values.

SETUP_DESIGN

The design is loaded and the family has been specified for the design.

DEVICE_SELECTION

The design has completed device selection (die and package). This corresponds to having successfully called the set_device command to set the die and package.

NETLIST_IMPORT

The design has imported a netlist.

COMPILE

The design has completed the compile command.

LAYOUT

The design has completed the layout command.

BACKANNOTATE

The design has exported a post-layout timing file (e.g. SDF).

PROGRAMMING_FILE

The design has exported a programming file (e.g. AFM).

Notes

1. Certain commands can only be used after Compile or Layout has been completed.
2. The is_design_state_complete command allows a script to check the design state before calling one of these state-limited commands.

Exceptions

1. The command will return an error if a design is not loaded.
2. The command will return an error if more than one argument is passed.
3. The command will return an error if the argument is not one of the valid values.

Example

The following code runs layout, but checks that the design state for layout is complete before calling backannotate.

```
layout –timing_driven
set bLayoutDone [ is_design_state_complete LAYOUT ]
if { $bLayoutDone  != 0 } {
  backannotate -name {mydesign_ba}  -format "SDF" -language
"verilog"
  }
}
```

See Also

```
compile
get_design_filename
get_design_info
is_design_loaded
is_design_modified
layout
set_design
set_device
```

layout (advanced options for the SX family)

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

Supported Family and Format

Family: SX

Format: Tcl

Syntax

```
layout [-timing_driven] [-incremental inc_mode] [-extended_run
ext_mode] [-effort_level enumber] [-timing_weight tnumber]
     where inc_mode = “on” | “off” | “fix” , ext_mode =”on” |
”off” , enumber is 25 to 500, tnumber is 10-150
```

new_design

The new_design command creates a new design.

Supported Family and Format

Family: All

Format: TCL

Syntax

```
new_design -name design_name -family family_name –path
pathname
```

Arguments

`-name design_name .`

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name.`

The Actel device family for which the deisgn is being targeted.

`-path path_name.`

The physical path of the directory in which the design files will be created.

Notes

You need all the 3 arguments for this command. This command will setup the Designer software for importing design source files.

Exceptions

None.

Example

Example 1: Creates a new ACT3 design with the name “test” in the current folder.

```
new_design -name "test" -family "ACT3" -path {.}
```

Example 2: These set of commands create a new design through variable substitution.

```
set desName “test
set famName “ACT3”
set path {d:/examples/test}
new_design -name $desName -family $famName -path $path
```

Example 3: Design creation and catch failures

```
if  { [catch { new_design -name $desName -family $famName
-path $path }] {
                Puts “Failed to create a new design”
                # Handle Failure
} else {
                puts “New design creation successful”
                # Proceed to Import source files
}
```

See Also

open_design, save_design, close_design, set_design

open_design

The open_design command opens an existing design into the Designer software.

Supported Family and Format

Family: All

Format: Tcl

Syntax

`open_design` ***`file_name`***

Arguments

file_name is the complete adb file path. The complete path is not provided then the directory is assumed to be the current working directory.

Notes

All previously open designs must be closed before opening a new design.

Exceptions

None.

Example

Example 1: Opens an existing design from the file “test.adb” in the current folder.

```
open_design {test.adb}
```

Example 2: Design creation and catch failures.

```
set designFile {d:/test/my_design.adb}
if  { [catch { open_design $designFile }] {
              Puts “Failed to open design”
              # Handle Failure
} else {
              puts “Design opened successfully”
              # Proceed to further processing
}
```

See Also

new_design, save_design, close_design

pin_assign

The pin_assign command assigns the pin, but does not fix its assignment.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

```
pin_assign [-nofix] -port <portname> -pin <pin number>
pin_assign  -port <port name> [-iostd <i/o standard>]
[-iothresh <i/othreshold>][-outload <output load>]
     [-slew <High | Low>][-res_pull <None | High | Low>]
```

Arguments

`-iostd`
Allows to set the I/O Standard

`-iothresh`
Allows to set the I/O Threshold

`-outload`
Allows to set the Output Load, also called Loading for some families

`-slew`
Allows to set the Slew

`-res_pull`
Allows to set the Resistor Pull, also called Power Up State for some families.

Notes

Must use pin_commit after this command.

Exceptions

None.

pin_commit

The pin_commit command saves the pin assignments to the .adb file.

Supported Family and Format

Family: ALL

Format: Tcl

Syntax:

```
pin_commit
```

Arguments

None.

Notes

This is needed after all pin commands to save changes.

Exceptions

None.

Example

Example 1:

pin_commit

See Also

pin_fix, pin_unfix, pin_assign, pin_unassign

pin_fix

This is equivalent to fixing a pin assignment.

Supported Family and Format

Family: All

Format:Tcl

Syntax:

```
pin_fix -port port_name
```

Arguments

`-port` *`port_name`*

specifies the port name for which the pin needs to be fixed at its placed location.

Notes

Fixed pins cannot be moved during place-and-route. Must use pin_commit after this command.

Exceptions

None.

Example

Example 1:

Pin_fix –port {clk}

Pin_commit

See Also

pin_commit, pin_unfix, pin_assign, pin_unassign

pin_fix_all

The pin_fix_all commands fixes all the placed pins on the device.

Supported Family and Format

Family: All

Format: TCL

Syntax:

```
pin_fix_all
```

Arguments

```
None.
```

Notes

Must use pin_commit after this command exceptions

Example

Example 1:

Pin_fix_all

Pin_commit

See Also

pin_fix, pin_unfix, pin_commit, pin_unassign

pin_unassign

The pin_unassign command unassigns a specific pin.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

`pin_unassign -port port_name`

Arguments

`-port port_name`

specifies the port for which the pin must be unassigned.

Notes

The unassigned pin location is now available for other ports. Must use pin_commit after this command.

Exceptions

Example

Example 1:

```
Pin_unassign –port “clk”
Pin_commit
```

See Also

pin_fix, pin_unfix, pin_commit, pin_unassign

pin_unassign_all

The pin_unassign_all command unassigns all the pins.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

pin_unassign_all

Arguments

None.

Notes

Now all the pin locations are available for assignment. Must use pin_commit after this command.

Exceptions

None.

Example

Example 1:

pin_unassign_all

pin_commit

See Also

pin_fix, pin_unfix, pin_commit, pin_unassign

pin_unfix

The pin_unfix command unfixing a pin assignment, allowing it to be moved during place-and-route.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

```
pin_unfix -port port_name
```

Arguments

`-port port_name`

specifies the port name that must be unfixed.

Notes

Pin_commit command must be used for this command to take effect.

Exceptions

None.

Example

Example 1:

Pin_unfix –port “rst”

Pin_commit

See Also

pin_fix, pin_commit, pin_assign, pin_unassign

report

The report command gives the user the ability to generate the Power report using TCL

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{plus}

Format: Tcl

Syntax:

```
report -type "power" -sortby "Power Values" | "Alphabetical" -
sortorder "Descending"| "Ascending"  -style "Hierarchical" -
opcond "Typical" -stat_pow "TRUE"|"FALSE" -dyn_pow
"TRUE"|"FALSE" -domains "TRUE"|"FALSE" -annotated_pins
"TRUE"|"FALSE" -min_ratio "number" -max_depth "number" -
min_power "number mW"  {.\report_name.rpt}
```

Arguments

`-type "power"`

Specifies that the type for the report to be generated is a power report

`?-sortby "Power Values" | "Alphabetical"?`

Specifies the method of sorting the values in the report

`?-sortorder "Descending"| "Ascending"?`

Specifies the sort order of the values in the report

`?-style "Hierarchical" ?`

Specifies the style of displaying the results in the report

`?-opcond "Typical" ?`

Specifies what operating conditions to be used

`?-stat_pow "TRUE"|"FALSE"?`

Specifies whether to include the Static Power value in the report.

`?-dyn_pow "TRUE"|"FALSE"?`

Specifies whether to include the Dynamic Power in the report.

`?-domains  "TRUE"|"FALSE"?`

Specifies whether to include the modifies domains into the power report

`?-annotated_pins  "TRUE"|"FALSE"?`

Specifies whether to include the annotated pins into the report or not

`?-min_ratio "number" ?`

Specifies which block to be expanded based on the minimum power ratio of a block with respect to the overall power value.

`?-max_depth "number" ?`

Specifies the maximum hierarchy depth to be included in the report.

`?-min_power "number mW" ?`

Specifies which block to be expanded based on the minimum power value of a block.

`{.\report_name.rpt}`

Specifies the name and destination of the report

Notes

None

Exceptions

None

Example

```
report -type "power" -sortby "Power Values" -sortorder
"Descending" -style "Hierarchical" -opcond "Typical" -stat_pow
"TRUE" -dyn_pow "TRUE"
-domains "TRUE" -annotated_pins "TRUE" -min_ratio "10" -
max_depth "2"
 -min_power "2 mW"  {e:\SmartPower\report.rpt}
```

save_design

The save_design command saves the current design in Designer to a file.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
save_design filename
```

Arguments

The design is written to a file denoted by the variable *filename* as an ADB file.

Notes

If filename is not a complete path name, the ADB file is written into the current working directory.

Exceptions

None.

Example

Example 1: Saves the design to a file “test.adb” in the current folder.

```
save_design {test.adb}
```

Example 2: Save design and check if it saved successfully.

```
set designFile {d:/test/my_design.adb}
if  { [catch { save_design $designFile }] {
              Puts “Failed to save design”
              # Handle Failure
} else {
              puts “Design saved successfully”
              # Proceed to make further changes
}
```

See Also

open_design, close_design, new_design

set_design

This set_design command specifies the design name, family and path in which Designer will process the design. This step is absolutely required before importing the source files.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_design -name design_name -family family_name –path
path_name
```

Arguments

`-name design_name .`

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name.`

The Actel device family for which the design is being targeted.

`-path path_name.`

The physical path of the directory in which the design files will be created.

Notes

You need all 3 arguments for this command to setup your design.

Example

Example 1: Sets up the design and checks if there are any errors

```
set_design -name "test" -family "Axcelerator" -path {.}
set desName “test
set famName “ACT3”
set path {d:/examples/test}

if  { [catch { set_design -name $desName -family $famName -
path $path }] {
              Puts “Failed setup design”
```

```
                # Handle Failure
  } else {
                puts “Design setup successful”
                # Proceed to Import source files
  }
```

See Also

new_design, set_device

set_device

The set_device command specifies the type of device and its parameters.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_device Option1 ?Option2 Option3 …?

   Options:
             ?-family family_name?
             ?-die die_name?
             ?-package package_name?
             ?-speed speed_grade?
             ?-voltage voltage?
             ?-voltrange volt_range?
             ?-temprange temp_range?
             ?-pci yes|no?
             ?-jtag yes|no?
             ?-probe yes|no?
             ?-itol 3.3|5.0?
             ?-io_trip pci|ttl?
             ?-trst yes|no?
          ?-scope “session”|…?
          ?-iostd “PCIX|…?
```

Arguments

`-family` *`family_name`*

Specifies the name of the FPGA device family.

`-die` *`die_name`*

Specifies the part name.

`-package` *`package_name`*

Specifies the selected package for the device.

-speed *speed_grade*

Specifies the speed grade of the part.

`-voltage voltage`

Specifies the core voltage of the device.

`-voltrange volt_range`

Specifies the voltage range to be applied for the device. It is generally MIL, COM and IND denoting Military, Commercial and Industrial respectively.

-temprange temp_range

Specifies the voltage range to be applied for the device. It is generally MIL, COM and IND denoting Military, Commercial and Industrial respectively.

`-pci yes|no`

Specified if the device needs to configure the IO for PCI specification.

`-jtag yes|no`

Specifies if pins need to be reserved for JTAG.

`-probe yes|no`

Specifies if the pins need to be preserved for probing.

`-trst yes|no`

Specifies if the pins need to be reserved for JTAG test reset.

Notes

At least one option must be specified for this command. Some of the options may not apply for certain families that do not support the features.

Example

Example 1: Setting up a PA design.

```
set_device -die "APA075" -package "208 PQFP" -speed "STD" -
voltage "2.5" \
-jtag "yes" -trst "yes" -temprange "COM" -voltrange "COM"
```

See Also

new_design, set_design

set_defvar

The set_defvar command sets an internal variable in the Designer system.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
set_defvar variable value
```

Arguments

Variable must be a valid Designer internal variable and could be accompanied by an optional value. If the *value* is provided, the *variable* is set the *value*. If the *value* is null the *variable* is reset.

Notes

Must have at least one argument.

Exceptions

None.

Example

Example 1:

```
set_defvar “FOREMAT” “VHDL”
Sets the FORMAT internal variable to VHDL.
```

Example 2:

```
set variableToSet "DESIGN"
set valueOfVariable “VHDL”
set_defvar $variableToSet $valueOfVariable
```

These commands set the FORMAT variable to VHDL, shows the use of varables for this command.

See Also

get_defvar

smartpower_add_pin_in_domain

smartpower_add_pin_in_domain adds a pin into a Clock or Set domain.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC[plus]

Format: Tcl

Syntax:

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type
{clock} | {set} – domain_name {domain_name} -domain_type
{clock} | {set}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to be added to the domain

-pine_type *{clock}* | *{data}*

Specifies the type of the pin to be added. The pin added will be either a clock pin or a data pin.

`-domain_name {domain_name}`

Specifies the name of the domain to be added

`-domain_type {clock} | {set}`

Specifies the type of the domain to be added.

Notes

- The *domain_name* must be a name of an existing domain.
- The *pin_name* must be a name of a pin that exists in the design.

Exceptions

None

Example

Example 1: To a pin to an existing Clock domain

```
smartpower_add_pin_in_domain -pin_name { XCMP3/U0/U1:Y } -
pin_type {clock} –domain_name {clk1} -domain_type {clock}
```

Example 2: To add a data pin to an existing Set domain

```
smartpower_add_pin_in_domain -pin_name {XCMP3/U0/U1:Y}   -
pin_type {data} -domain_name {myset} -domain_type {set}
```

See Also

smartpower_remove_pin_of_domain

smartpower_commit

The smartpower_commit command saves the changes made to the Designer database.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC[plus]

Format: Tcl

Syntax:

```
Smart_power_commit
```

Arguments

None

Notes

None

Exceptions

None

Example

```
Smart_power_commit
```

See Also

smartpower_restore

smartpower_create_domain

The smartpower_create_domain creates a new clock or set domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASICplus

Format: Tcl

Syntax:

```
smartpower_create_domain -domain_type {clock}| {set} -
domain_name {domain_name}
```

Arguments

`-domain _type {clock}| {set}`

Specifies that the domain that is being created is either a clock domain or set domain

`-domain_name {domain_name}`

Specifies the domain name to be created.

Notes

The –domain_type must be either “clock” or “set”

The domain name must not be a name of an existing domain.

Exceptions

None

Example

```
smartpower_create_domain -domain_type {clock} -domain_name
{clk2}
smartpower_create_domain -domain_type {set} -domain_name
{myset}
```

See Also

smartpower_remove_domain

smartpower_remove_domain

The smartpower_remove_domain removes and existing domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC[PLUS]

Format: Tcl

Syntax:

```
smartpower_remove_domain -domain_type {clock}| {set} -
domain_name {domain_name}
```

Arguments

`-domain _type {clock}| {set}`

Specifies that the domain that is being removed is either a clock domain or a set domain

`-domain_name {domain_name}`

Specifies the domain name to be removed.

Notes

- The –domain_type must be either “clock” or “set”
- The domain name must be a name of an existing domain.

Exceptions

None

Example

```
smartpower_remove_domain -domain_type {clock} -domain_name
{clk2}
smartpower_remove_domain -domain_type {set} -domain_name
{myset}
```

See Also

smartpower_create_domain

smartpower_remove_pin_frequency

smartpower_remove_pin_frequency command removes the frequency of a pin

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^plus

Format: Tcl

Syntax:

```
smartpower_remove_pin_frequency -pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin for which the frequency will be removed

Notes

- The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.

Exceptions

None

Example

smartpower_remove_pin_frequency -pin_name {count8_clock}

See Also

smartpower_set_pin_frequency

smartpower_remove_pin_of_domain

smartpower_remove_pin_of_domain removes a clock pin or a data pin from a Clock or Set domain respectively.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC[plus]

Format: Tcl

Syntax:

smartpower_remove_pin_of_domain -pin_name {*pin_name*} -pin_type *{data}* | *{clock}* -domain_name {*domain_name*} -domain_type *{set}*| *{clock}*

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to be removed

`-pin_type {data} | {clock}`

Specifies the type of the pin to be removed .The pin could be either a data pin or a clock pin.

`-domain_name {domain_name}`

Specifies the domain name from which to delete the pin

`-domain_type {set}| {clock}`

Specifies the type of the domain from which a pin is being removed.

Notes

- The *pin_name* must be a name of a pin that already exists in the design
- The *domain_name* must be a name of an existing domain

Exceptions

None

Example

Example 1: Removes a pin from a Clock domain

```
smartpower_remove_pin_of_domain -pin_name {XCMP3/U0/U1:Y}
-pin_type {clock} -domain_name {clockh} -domain_type {clock}
```

Example 2: Removes a data pin from a Set domain

```
smartpower_remove_pin_of_domain -pin_name {count2_en}  -
pin_type
{data}  -domain_name {InputSet} -domain_type {set}
```

See Also

smartpower_add_pin_in_domain

smartpower_restore

The smartpower_restore command restores previous committed constraints

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{plus}

Format: Tcl

Syntax:

Smart_power_restore

Arguments

None

Notes

None

Exceptions

None

Example

Smart_power_restore

See Also

smartpower_commit

smartpower_set_domain_frequency

The smartpower_set_domain_frequency sets the frequency of a domain

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^plus

Format: Tcl

Syntax:

```
smartpower_set_domain_frequency –domain_type {clock}| {set} –
domain_name   {domain_name} –clock_freq {clock_frequency} –
data_freq {data_frequenc}
```

Arguments

`-domain _type {clock}| {set}`

Specifies that the domain to set the frequency for is either a Clock domain or a Set domain

`-domain_name {domain_name}`

Specifies the domain name

`-clock_freq {clock_frequency}`

Specifies the frequency values in positive decimal number. Units in MHz.

`-data_freq {data_frequenc}`

Specifies that data frequency of the domian

Notes

- The –domain_type must be either “clock” or “set”
- The domain name must be a name of an existing domain.
- The clock frequency must be a positive decimal number. Specifying the unit as part of the frequency value is optional. There must be a space between the frequency value and the unit. The clock frequency needs to be set only for the clock domains.
- The data frequency must be a positive decimal number. Specifying the unit as part of the data frequency value is optional. There must be a space between the data frequency value and the unit.

Exceptions

None

Example

Example 1: Setting the clock frequency and the data frequency of a clock domain.

```
smartpower_set_domain_frequency –domain_type {clock} –
domain_name {clk1} –clock_freq {32} or {30 MHZ} –data_freq {3}
or {3 Mhz}
```

Example 2: Setting the data frequency of a set domain.

```
smartpower_set_domain_frequency –domain_type {set} –
domain_name {set1} –data_freq {10}.
```

See Also

smartpower_create_domain

smartpower_remove_domain

smartpower_set_pin_frequency

The smartpower_set_pin_frequency command sets the frequency of a pin.

Supported Family and Format

Family: Axcelerator, ProASIC, and ProASIC^{plus}

Format: Tcl

Syntax:

smartpower_set_pin_frequency -pin_name {*pin_name*} -pin_freq {*frequency_value*}

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin for which the frequency will be set

`-pin_freq {frequency_value}`

Specifies the values of the frequency. The frequency can be any positive decimal number.

Units in MHz

Notes

- The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.
- When specifying the unit, a space must be between the frequency value and the unit.

Exceptions

None

Example

```
smartpower_set_pin_frequency -pin_name {count8_clock} -
pin_freq {100}
```

See Also

smartpower_remove_pin_frequency

timer_add_clock_exception

The timer_add_clock_exception adds an exception to or from a pin with respect to a specified clock.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_add_clock_exception -clock clock_name -pin pin_name -dir {from}|{to}
```

Arguments

`-clock` *`clock_name`*

Specifies the clock name.

`-pin` *`pin_name`*

Specifies the exception on the pin in a synchronous network that should be excluded from the specified clock period.

`-dir {from}|{to}`

Specifies direction. <Need to explain more>

Notes

None.

Exceptions

None.

Example

Example 1: Adding a clock exception from the pin reg_q_a_10_/U0:CLK with respect to the clock clk.

```
timer_add_clock_exception -clock {clk} -pin
{reg_q_a_10_/U0:CLK} -dir {from}
```

Example 2: Adding a clock exception to the pin reg_q_a_10_/U0:E with respect to the clock clk.

```
timer_add_clock_exception -clock {clk} -pin
{reg_q_a_10_/U0:E} -dir {to}
```

timer_add_pass

The timer_add_pass command adds the pin to the list of pins for which the path must be shown passing through, in the timer.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_add_pass
    -pin pin_name
```

Arguments

`-pin` *`pin_name`*

Specifies the name of the pin to be included for displaying the timing path through it.

Notes

Setting a pass on a module pin enables you to see a path through individual pins.

Example

Example 1: Adding pass through the pin reg_q_a_0_:CLK

```
timer_add_pass -pin {reg_q_a_0_:CLK}
```

Example 2: Adding pass through a clear pin reg_q_a_0_:CLRin the design

```
timer_add_pass -pin {reg_q_a_0_:CLR}
```

See Also

timer_add_stop

timer_add_stop

The timer_add_stop command adds the specified pin to the list of pins through which the paths will not be displayed in the timer.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_add_stop -pin pin_name
```

Arguments

`-pin` *`pin_name`*

specifies the name of the pin to be excluded from displaying the path.

Notes

Without stop points, you see all the paths from pad to pad in the design. If you do not want to see the paths going through registers clock pins, you could specify these as stop points. The path going through those pins would not be displayed.

Exceptions

None.

Example

Example 1: Adding stop to the pin a<2>

timer_add_stop -pin {a<2>}

Example 2: Adding a stop to a clock and a clear pin in the design

timer_add_stop -pin {reg_q_a_0_:CLK}
timer_add_stop -pin {reg_q_a_0_:CLR}

See Also

Timer_add_pass

timer_commit

The timer_commit command saves the changes made to constraints into the Designer database.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_commit
```

Arguments

None.

Notes

None.

Exceptions

None.

Example

Example 1:

timer_commit

timer_get_path

The timer_get_path command obtains the path information from the timer and reports it to the log window.

Supported Family and Format

Family: All

Format: Tcl

Syntax:

```
timer_get_path -from source_pin -to destination_pin
    ?-exp no | yes ?
    ?-sort actual | slack?
    ?-order  long | short ?
    ?-case  worst | typ | best ?
    ?-maxpath maximum_paths ?
    ?-maxexpath maximum_paths_to_expand?
    ?-mindelay minimum_delay?
    ?-maxdelay maximum_delay?
    ?-breakatclk  no | yes ?
    ?-breakatclr  yes | no ?
```

Arguments

`-from source_pin`
specifies the source pin of the path.

`-to destination_pin`
specifies the destination pin for the path.

`?-exp no | yes ?`
specifies if the path needs to be expanded.

`?-sort actual | slack?`
specifies if the paths need to be sorted based on actual delay or the slack value.

`?-order  long | short ?`
specifies if the maximum list size is based on longest or shortest paths.

`?-case  worst | typ | best ?`
specifies if the report must consider timing values for the worst, typical or best cases.

`?-maxpath maximum_paths ?`
specifies the maximum number of paths to be reported.

`?-maxexpath` *`maximum_paths_to_expand`*`?`
specifies if number of maximum paths to be expanded.

`?-breakatclk  no | yes ?`
specifies if the paths must be broken at the register clock pins

`?-breakatclr  yes | no ?`
specifies if the paths must be broken at the register clear pins

Notes

None.

Exceptions

None.

Example

Example1:

```
timer_get_path -from "headdr_dat<31>" \
    -to  "u0_headdr_data1_reg/data_out_t_31/U0:D" \
    -case typ  \
    -exp "yes" \
    -maxpath "100" \
    -maxexpapth "10"
```

timer_get_clock_actuals

The timer_get_clock_actuals finds and reports the actual clock frequency when timer is initiated.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_get_clock_actuals -clock “name”
```

Arguments

List supported arguments here with an explanation of each argument.

Notes

The actual clock frequency is reported in the log window.

Exceptions

None.

Example

Example 1: Reports the actual clock frequency on the clock clk1

```
timer_get_clock_actuals -clock clk1
```

timer_get_clock_constraints

The timer_get_clock_constraints? <No idea what this does. No info anywhere>

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_get_clock_constraints -clock clock_name
```

Arguments

`-clock` *`clock_name`*

specifies the clock for which the constraint needs to be reported.

Notes

None

Exceptions

None.

Example

Example 1: Reports the clock constraint on the clock clk1

```
timer_get_clock_constraints -clock clk
```

timer_get_maxdelay

The timer_get_maxdelay command obtains maximum delay constraint between 2 pins of a path.

Supported Family and Format

Family: All

Format: Tcl

Syntax

`timer_get_maxdelay -from source_pin -to destination_pin`

Arguments

`-from source_pin`

Specified the source pin of the path.

`-to destination_pin`

Specifies the destination pin of the path.

Notes

You can use the following macros in the command:

`$in()`

to specify all inputs.

`$out()`

to specify all output pins.

`$reg(clock_name)`

to specify all registers relates to *clock_name*..

Exceptions

None.

Example

Example 1: Get the max delay constraint from all registers of clk166 to all outputs

```
timer_get_maxdelay -from {$reg(clk166)[*]} -to {$out()[*]}
```

Example 2: Get the max delay constraint from the pin clk166 to the pin reg_q_a_9_/U0:CLK

```
timer_get_maxdelay -from {clk166} -to {reg_q_a_9_/U0:CLK}
```

Example 2: Get max delay constraint from all inputs to all registers of clock166 and also check for errors in the command.

```
if [ catch {timer_get_maxdelay -from {$in()[*]} -to
{$reg(clk)[*]}} ] {
  puts "Error getting max_delay information"
} else {
  puts "Successfully obtained max_delay information"
}
```

See Also

timer_set_maxdelay

timer_get_path_constraints

The timer_get_path_constraints reports the path constraints that were set as max delay in the timer.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_get_path_constraints
```

Arguments

None.

Notes

If no max delay constraints are set this command will not report any delay values. <Can report “No valid path constraints found”>. The information is displayed in the log window.

Exceptions

None.

Example

```
Timer_get_path_constraints.
```

timer_remove_clock_exception

The timer_remove_clock_exception command removes the previously set clock constraint.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_remove_clock_exception -clock clock_name -pin pin_name -
dir {from}|{to}
```

Arguments

`-clock` *`clock_name`*

Specifies the clock name.

`-pin` *`pin_name`*

Specifies the exception to be removed on the pin in a synchronous network.

`-dir {from}|{to}`

Specifies the direction <Need to explain more>

Notes

None.

Exceptions

None.

Example

Example 1: Removing a clock exception from the pin reg_q_a_10_/U0:CLK with respect to the clock clk.

```
timer_remove_clock_exception -clock {clk} -pin
{reg_q_a_10_/U0:CLK} -dir {from}
```

Example 2: Removing a clock exception to the pin reg_q_a_10_/U0:E with respect to the clock clk.

```
timer_remove_clock_exception -clock {clk} -pin
{reg_q_a_10_/U0:E} -dir {to}.
```

See Also

timer_add_clock_exception

timer_remove_pass

The timer_remove_pass command removes the path pass constraint that has been previously entered.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_remove_pass -pin pin_name
```

Arguments

`-pin` *`pin_name`*

specifies the pin for which the path pass constraint must be removed.

Notes

None.

Exceptions

None.

Example

Example1 : removes the pass constraint from the clock pin reg_q_a_0_:CLK.

```
timer_remove_pass -pin {reg_q_a_0_:CLK}
```

See Also

timer_add_pass

timer_remove_stop

The timer_remove_stop command removes the path stop constraint on the specified pin that has been previously entered.

Supported Family and Format

Family: All

Format: TCL

Syntax

```
timer_remove_stop -pin pin_name
```

Arguments

-pin *pin_name*, specifies the pin for which the path stop constraint must be removed..

Notes

- Export of script writes the constraint wrong with timer_remove_pass instead of timer_remove_stop

Exceptions

None.

Example

Example 1: Removes the path stop constraint on the clear pin reg_q_a_0_:CLR.

timer_remove_stop -pin {reg_q_a_0_:CLR}

See Also

Timer_add_stop

timer_restore

The `timer_restore` command restores previous committed constraints.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_restore
```

Arguments

None.

Exceptions

None.

timer_setenv_clock_freq

The timer_setenv_clock_freq command sets a required clock frequency for the specified clock, in MHz.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_setenv_clock_freq -clock clock_name -freq
frequency_value
```

Arguments

`-clock` *`clock_name`*

Specifies the clock for which the constraint is intended.

`-freq` *`frequency_value`*

Specifies the frequency in MHz.

Notes

None.

Exceptions

None.

Example

Example 1: Sets a clock frequency of 100MHz on the clock clk1

```
timer_setenv_clock_freq -clock {clk1} -freq 100.00
```

timer_setenv_clock_period

The timer_setenv_clock_period command sets the clock period constraint on the specified clock.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_setenv_clock_period -clock clock_name ?-unit
{ns}|{ps}? -period period_value
```

Arguments

`-clock` *`clock_name`*

specifies the clock name for which the constraints in intended.

`?-unit “ns”|”ps”?`

optional argument specifies the unit for the clock period constraint. Default is “ns”

`-period` *`period_value`*

specifies the period in the specified unit.

Notes

None.

Exceptions

None.

Example

Example 1: Sets a clock period of 2.40ns on the clock clk1

```
timer_setenv_clock_period -clock {clk1} -unit {ns} -period
2.40
```

timer_set_maxdelay

The timer_set_maxdelay adds a maximum delay constraint for the path.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_set_maxdelay -from source_pin -to destination_pin ?-unit
{ns}|{ps}? -delay delay_value
```

Arguments

`-from source_pin`

Specifies the source pin of the path.

`-to destination_pin`

Specifies the destination pin of the path.

`?-unit “ns”|”ps”?`

Specifies if the delay unit.

`-delay delay_value`

Specifies the actual delay value for the path.

Notes

You can use the following macros in the command:

```
$in(), to specify all inputs.
$out(), to specify all output pins.
$reg(clock_name), to specify all registers relates to
clock_name.
```

Exceptions

None.

Example

Example 1: Set 20ns delay from all registers of clk166 to all outputs

```
timer_set_maxdelay -from {$reg(clk166)[*]} -to {$out()[*]} -unit {ns} -delay 20.00
```

Example 2: Set delay value of 30ns from all inputs to all outputs

timer_set_maxdelay -from {$in()[*]} -to {$out()[*]} -unit {ns} -delay 30.00

timer_remove_all_constraints

The `timer_remove_all_constraints` command removes all the timing constraints that have already been entered in the Designer system.

Supported Family and Format

Family: All

Format: Tcl

Syntax

```
timer_remove_all_constraints
```

Arguments

None.

Notes

None.

Exceptions

None.

Example

Example 1: Remove all constraints and commit the changes.

```
timer_remove_all_constraints
timer_commit
```

See Also

timer_commit

Device Programming

Device programming begins with generating your programming files. For more information on programming hardware, see http://www.actel.com/products/tools/prog.asp.

Generating Programming Files

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

Programmer	Antifuse Programming File	Flash Programming File
Flash Pro	N/A	.stp
Silicon Sculptor I	.afm (Non-Axcelerator families)	.bit
Silicon Sculptor II	.afm	.bit .stp (Windows only)

FlashLock

Actel's ProASIC and ProASIC^{PLUS} devices contain FlashLock circuitry to lock the device by disabling the programming and readback capabilities after programming. Care has been taken to make the locking circuitry very difficult to defeat through electronic or direct physical attack.

FlashLock has three security options: No Lock, Permanent Lock, and Keyed Lock.

No Lock

Creates a programming file which does not secure your device.

Permanent Lock

The permanent lock makes your device one time programmable. It cannot be unlocked by you or anyone else.

Keyed Lock

Within each ProASIC and ProASICPLUS device, there is a multi-bit security key user key. The number of bits depends on the size of the device. The tables below show the key size of different ProASIC and ProASIC^{PLUS} devices, respectively. Once secured, read permission and write permission can only be enabled by providing the correct user key to first unlock the device. The maximum security key for the device is shown in the dialog box.

Key Size of ProASIC Devices

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

Key Size of ProASIC^{PLUS} Devices

Device	Key Size (bits)	Key Size (Hex)
APA075	79 Bits	20
APA150	79 Bits	20
APA300	79 Bits	20
APA450	119 Bits	30
APA600	167 Bits	42
APA750	191 Bits	48
APA1000	263 Bits	66

Programming the Security Bit

Two device programmers, Silicon Sculptor and Flash Pro, are available for ProASIC and ProASIC^{PLUS} devices. If the programming file contains the security key, by default the Silicon Sculptor and Flash Pro programming software automatically enables the "secure" option and

programs the security key. You can turn this off, should you decide not to program using the security key.

Please refer to the application note "Implementation of Security in Actel's ProASIC and ProASIC^{PLUS} Flash-Based FPGAs" for more details.

Generating Bitstream and STAPL files

Bitstream allows you to generate a bitstream or STAPL file for ProASIC and ProASIC^{PLUS} devices. Please consult the Program Files table to find out which file type you should choose.

To generate a bitstream or STAPL file:

1. In the **Tools** menu, click **Bitstream** or click the Bitstream button in the Design Flow window.
2. Select **Bitstream** or **STAPL** from the File Type drop-down list box.
3. **FlashLock**. Select one of the following options:
    - No Locking: Creates a programming file which does not secure your device.
    - Use Keyed Lock: Creates a programming file which secures your device with a FlashLock key. The maximum security key for the device is shown in the dialog box. The maximum security key for the device is shown in the dialog box.
    - Use Permanent Lock: Creates a one-time programmable device.
4. Click **OK**. Designer validates the security key and alerts you to any concerns.

Note: The bitstream file header contains the security key.

Generating a fuse file

Fuse allows you to generate a programming file for your Actel Antifuse devices. Fuse files work with Actel's Silicon Sculptor programmers. (For Axcelerator families, you must use the Silicon Sculptor II programmer.)

To generate a fuse programming file:

1. In the **Tools** menu, click **Fuse** or click the Fuse button in the Design Flow window.

2. **File Type**. Select the appropriate file type in the File Type pull-down menu. Select "AFM-APS2" if you are using Silicon Sculptor programmer.
3. Silicon Signature (Optional): Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are "0" through "9," and "a" through "f."

- **Output filename:** Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box. Click Browse to change the directory. Do not add a file extension or suffix to the file name. The Designer software automatically adds the extension to the programming file name when you specify the programming format.
- **Generate Probe File Also:** This option automatically generates a .prb file for use with Silicon Explorer
- **Disable clamping diode for unused I/O pins:** (SX-A and eX families). Check box to disable clamping diode.
- **Use the JTAG Reset Pull-up Resistor**: (Axcelerator family) Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).
- **Use the Global Set Fuse**: (Axcelerator family) Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option id used, all flip-flops are set to '1' at power up.

2. Click **OK** when finished to save the file.

Generating prototype files

When designing for RTAX-S, you can use the Axcelerator family of devices for prototyping. Please refer to the application note, Prototyping RTAX-S Using Axcelerator Devices for more information.

To generate prototype files:

1. From the **Tools** menu, click **Generate Prototype**. In the Generate Prototype Files dialog box, make the following selections:
2. Silicon Signature (Optional). Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are “0” through “9,” and “a” through “f.”

- **Output filename**. Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box. Click Browse to change the directory. Do not add a file extension or suffix to the file name. The Designer software automatically adds the extension to the programming file name when you specify the programming format.
- **Generate Probe File Also**. This option automatically generates a .prb file for use with Silicon Explorer
- **Use the JTAG Reset Pull-up Resistor**: Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).
- **Use the Global Set Fuse**: Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option id used, all flip-flops are set to '1' at power up.

2. Click **OK**. The AFM file is generated.

Contacting Actel

Actel Headquarters

Actel Corporation is a supplier of innovative programmable logic solutions, including field-programmable gate arrays (FPGAs) based on antifuse and flash technologies, high-performance intellectual property (IP) cores, software development tools and design services targeted for the high-speed communications, application-specific integrated circuit (ASIC) replacement, and radiation-tolerant markets.

Address:	955 East Arques Avenue Sunnyvale, CA 94086
Phone:	(888) 99-ACTEL

Technical Support

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

Visit Tech Support Online

For 24-hour support resources, visit Actel Technical Support at http://www.actel.com/custsup/search.html.

Contacting Technical Support

Contact us with your technical questions via e-mail or by phone. Also, if you have design problems, you can e-mail your design files to receive assistance. When sending your request to us, please be sure to include your full name, company name, and telephone number.

E-mail:	tech@actel.com
Telephone (In U.S.):	(408) 522-4460 (800) 262-1060
Telephone (Outside the US):	Contact a local sales office

Customer Service

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

From	Call
Northeast and North Central U.S.A.	(408) 522-4480
Southeast and Southwest U.S.A.	(408) 522-4480
South Central U.S.A.	(408) 522-4434
Northwest U.S.A.	(408) 522-4434
Canada	(408) 522-4480
Europe	(408) 522-4252 or +44 (0) 1276 401500
Japan	(408) 522-4743
From the rest of the world	(408) 522-4743

Sales

For a list of our sales offices, please see http://www.actel.com/contact/offices/index.html.

Documentation Feedback

Actel Corporation strives to produce the highest quality online Help and printed documentation. We want to help you learn about our products. We welcome your feedback. Please send your comments to documentation@actel.com.

Index