Designer v8.4 User's Guide

Actel®
POWER MATTERS

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 5-02-9122-22

Release: July 2008

Trademarks

Table of Contents

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.

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

To implement your design:

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

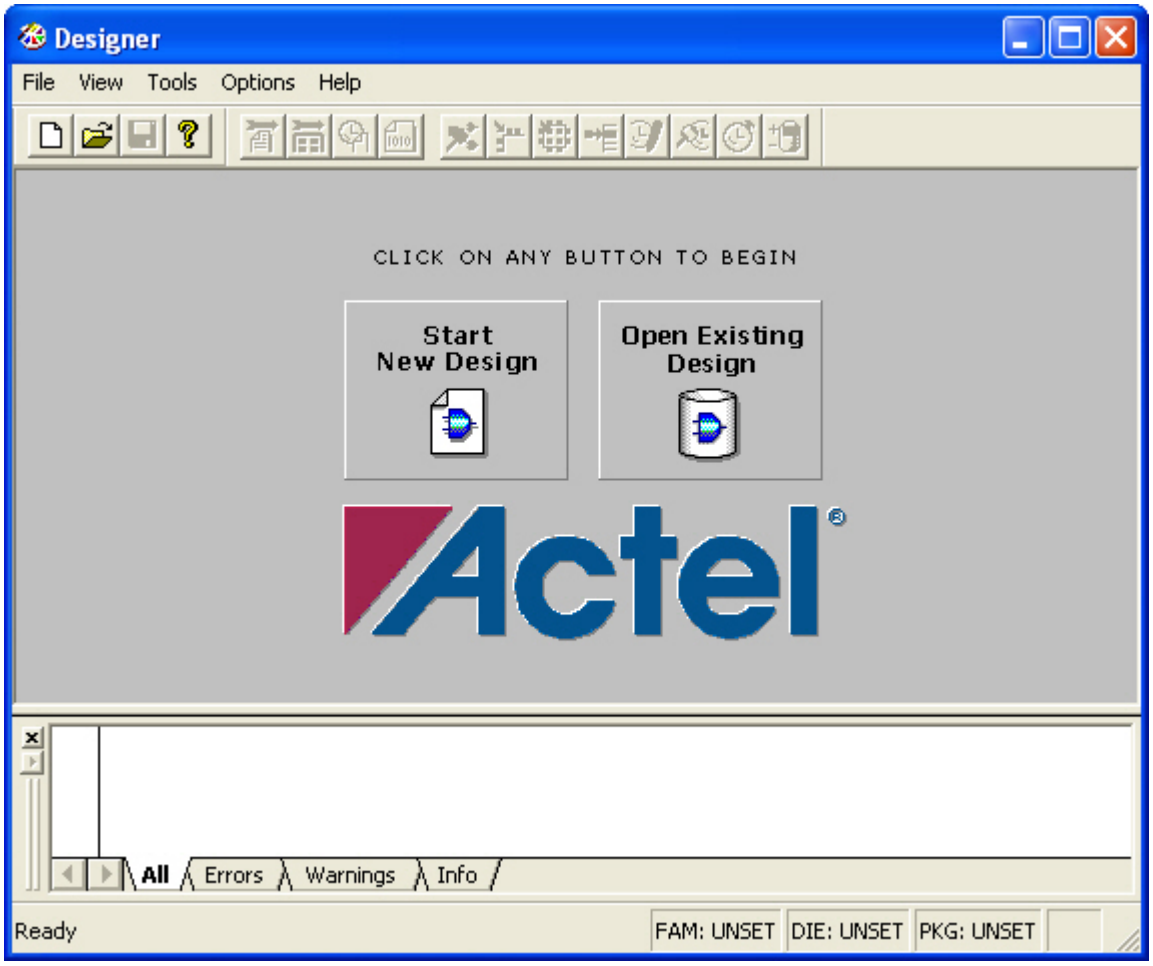
Tool	Function	Supported Families
PinEditor	Package-level floorplanner and I/O attribute editor	All
ChipPlanner	Logic viewer, placement- and floorplanning tool	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A
ChipEditor	Logic viewer and placement tool	SX, MX, 3200DX, ACT3, ACT2, ACT1
NetlistViewer	Design schematic viewer	All
SmartPower	Power analysis tool	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A
SmartTime and Timer	Static timing analysis and constraints editor (SmartTime only)	All
I/O Attribute Editor	Edit I/O attributes, layout	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

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

Starting Designer

To start Designer from Libero IDE:

In the Design Flow window, click **Designer Place-and-Route**.



Starting a New Design

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

Starting a new design creates an Actel ADB file. ADB files are proprietary Actel project files.

To start a new design:

1. Click **Start New Design** in the Designer main window, or from the **File** menu, choose **New**. This displays the Setup Design dialog box.
   Note: The Actel ADB file memory requirement is equivalent to 2x the size of the ADB file. If your computer does not have 2x the size of your ADB file's memory available, please make memory available on your hard drive.
2. Setup Design:
   - Enter a **Design Name**. The design name is used in reports and as the default name when saving or exporting files. The Design Name field is disabled when you open Designer place-and-route software from the Libero IDE. If you wish to change your design name, from the **File** menu, choose **Save As**, and save your file outside of the Libero folder structure.
   - If you are creating a Designer Block, select the **Enable Designer Block creation** checkbox. Designer Blocks are useful if you wish to duplicate design elements. Once you place-and-route and publish your Designer Block you can instantiate the block in as many designs as you like. Note that you cannot program your design in Designer Block mode, only publish the Designer Block files.
   - Select an **Actel Family** from the drop-down menu list.
   - Specify a **Working Directory**. Click **Browse** to locate a directory. The Working Directory field is disabled when you open Designer place-and-route software from the Libero IDE. If you wish to change your working directory, you must change your filename. To do so, from the **File** menu, select **Save As**, and save your file outside of the Libero folder structure.
3. Click **OK**. The Designer custom design flow window appears. All tools and commands are activated.

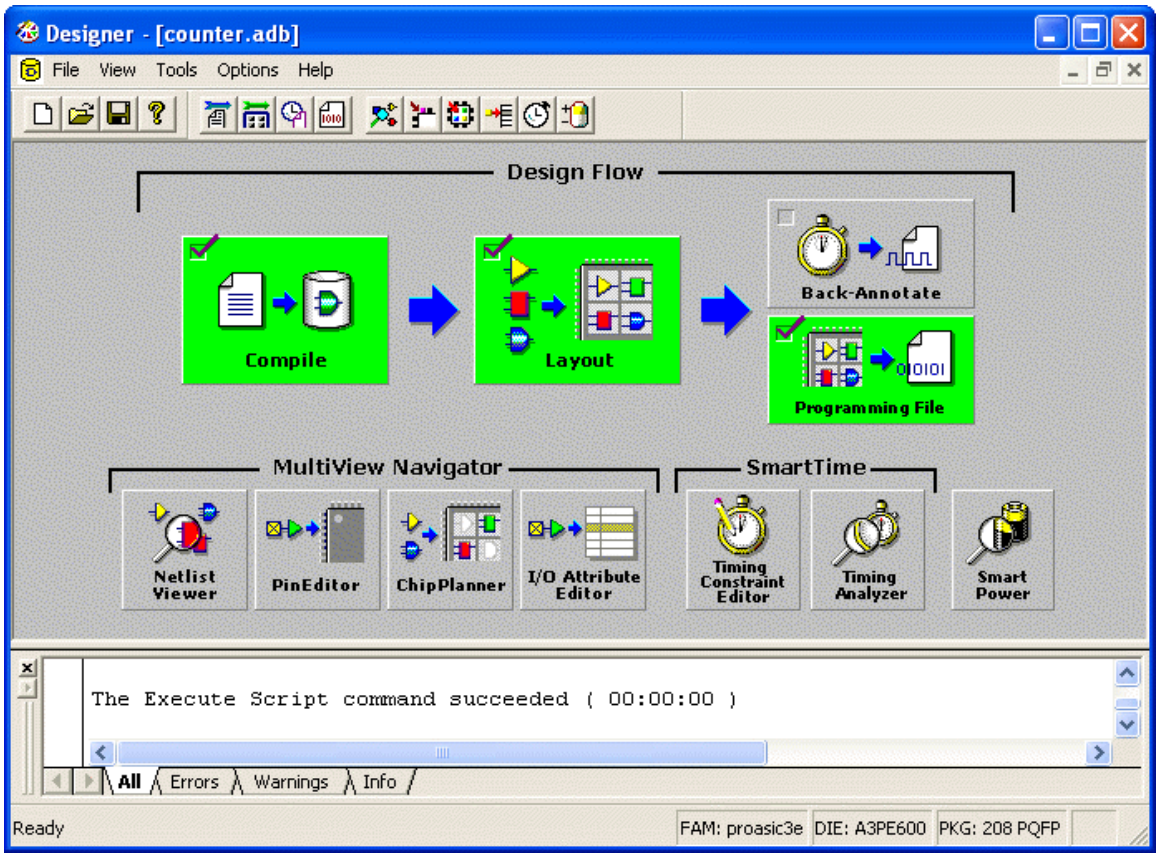


Figure 1 · Designer:New Design

Opening an Existing Design

To open an existing design:

1. Click **Open Existing Design** or from the **File** menu, choose **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**. The Designer 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.

See Also

Starting Designer

Starting a new design

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 the Actel Technical Support department 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 the 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 may receive 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.

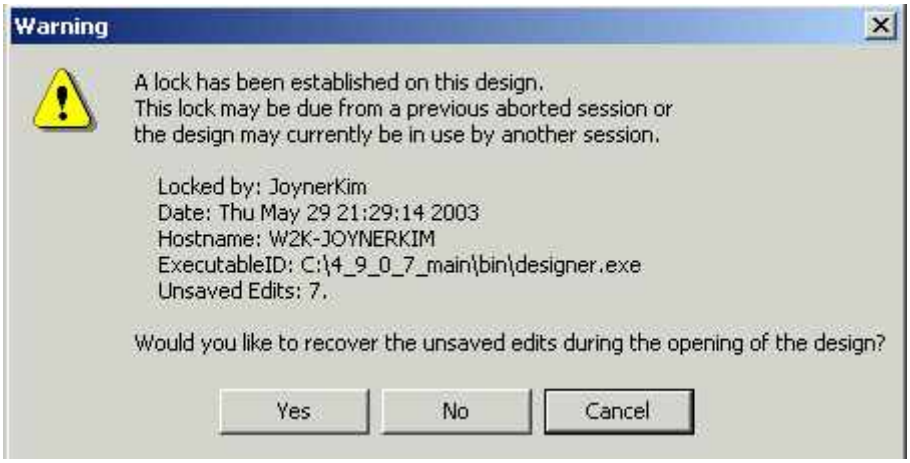


Figure 2 · Warning: Locked File

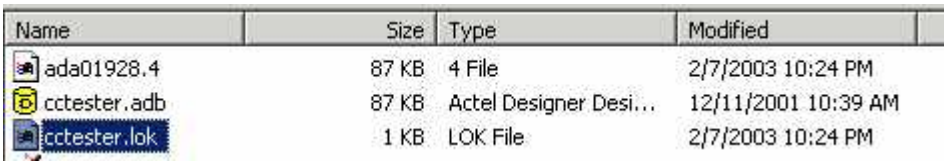
Error

When opening a design, you may receive an error that the file cannot be opened because the lock file is old. You cannot recover any unsaved edits.

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.



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, choose **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, choose **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, choose **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.

About Your Installation

To display information about your license:

From the **Start** menu, choose **Programs**, and then click the **Actel software folder** and select About Your Installation. The software displays your complete license configuration, all Actel-installed software and versions, as well as your HostID and disk volume serial number.

You can also select **License Details** from the **Help** menu in Designer to view your license information.

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, choose **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.
9. 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, choose **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, select **Check for Software Updates**.

Note: This feature requires an internet connection.

Proxy

A Proxy improves access to the Actel server. From the **File** menu, choose **Preferences**.

Click the **Proxy** tab to set your Proxy preferences.

To enable the proxy:

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

File association (PC only)

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.

You must have rights to modify the HKEY_CLASSES_ROOT of the registry of the machine to set Designer file association. If you do not have rights to modify the registry, Designer ignores your settings.

*To associate *.adb files with the Designer application:*

1. From the **File** menu, choose **Preferences**.
2. Select **Check the default file association (*.adb) at startup** check box to associate ADB files with the Designer application. Clear 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	Color
Errors	Red
Warnings	Light Blue
Informational	Black
Linked	Dark Blue

The default preference is to Clear the log window automatically. This clears the Designer log window each time you close or open a new design in Designer. Clear the box if you want Designer to leave the log information after you close a design.

PDF Reader (UNIX only)

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

Web Browser (UNIX only)

Specify the default web browser you wish to use on the UNIX platform. The web browser displays the online help.

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, choose **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 the 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. Select the **Reserve JTAG** check box to reserve the JTAG pins "TDI," "TMS," "TCK," and "TDO" during layout.
3. Select the **Reserve JTAG Reset** check box to reserve the JTAG reset Pin "TRST" during layout.
4. Select the **Reserve Probe** check 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.
5. 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.
6. 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 SmartTime, 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
- TGrade1
- TGrade2
- Custom

Consult the Actel Data Sheet, available at http://www.actel.com/techdocs/ds/, 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.

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

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 are adequate.

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

Voltage

Select a voltage:

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

If you select Custom, you may choose from Best > Typical > Worst in the drop-down menu.

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.

RTSX-S and RTAX-S ONLY - Radiation Derating

This option is only available for RTSX-S and RTAX-S devices. The valid range is integer values from 0 to 300, and the units are in KRads. Modifying this selection impacts the timing derating in SmartTime and back-annotating SDF files, so when you modify this value, you must extract a new SDF file from Designer and re-evaluate the timing of your design. It does not affect the device configuration.

Changing Design Name and Family

Design name and family are set when you create a new design. However, you can change this information for existing designs. If you change the family, you must re-import the netlist. 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 in Designer:

1. In the Designer **Tools** menu, choose **Setup Design**. This displays the Setup Design dialog box.
2. Specify the design name and family.
3. Click **OK**. Refer to the Actel datasheet for your device for family specifications.

You may wish to migrate your SX device to an SX-A device. The SX to SX-A compatible family change option is available in the Device Selection wizard.

To migrate your SX design to SX-A:

1. Open your SX design.
2. From the Designer **Tools** menu, choose **Device Selection**.
3. Select **SXA** from the **Change to** drop-down menu, and proceed in the Device Selection Wizard to complete the migration. You must re-compile and layout your design to run the Designer User Tools.

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 device information for existing designs:

1. In the **Tools** menu, choose **Device Selection**. The Device Selection Wizard appears.
2. Select Die, Package, and Speed Grade and click **Next**. (You must select a 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.

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 (1.2µm) to an A1020B (1.0µm) 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 Source Files

Source files include your netlist and constraint files.

Source files are files created by outside tools that will be tracked (audited) to better coordinate the design changes. If you wish, you may import some files as auxiliary files. Auxiliary files are not audited, but you do not have to re-compile your design after you import them.

Source Files	File Type Extension	Family
EDIF	*.ed*	All
Verilog	*.v	All
VHDL	*.vhd	All
Actel ADL Netlist	*.adl	All, except ACT1, ACT2, ACT3
Criticality	*.crt	ACT1, ACT2, ACT3, MX, 1200XL, DX
ProASIC Constraint File	*.gcf	ProASICPLUS
Physical Design Constraint File	*.pdc	IGLOO, Fusion, ProASIC3 and Axcelerator
Synopsys Constraint File	*.sdc	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, SX-S, SX-A, and eX
CoreConsole Database File	*.cdb*	IGLOO, Fusion, and ProASIC3; Designer uses the CDB file to place the CoreMP7 cores and Designer blocks.

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. From the **File** menu, choose **Import Source Files**. This displays the Import Source Files dialog box, as shown in the figure below.

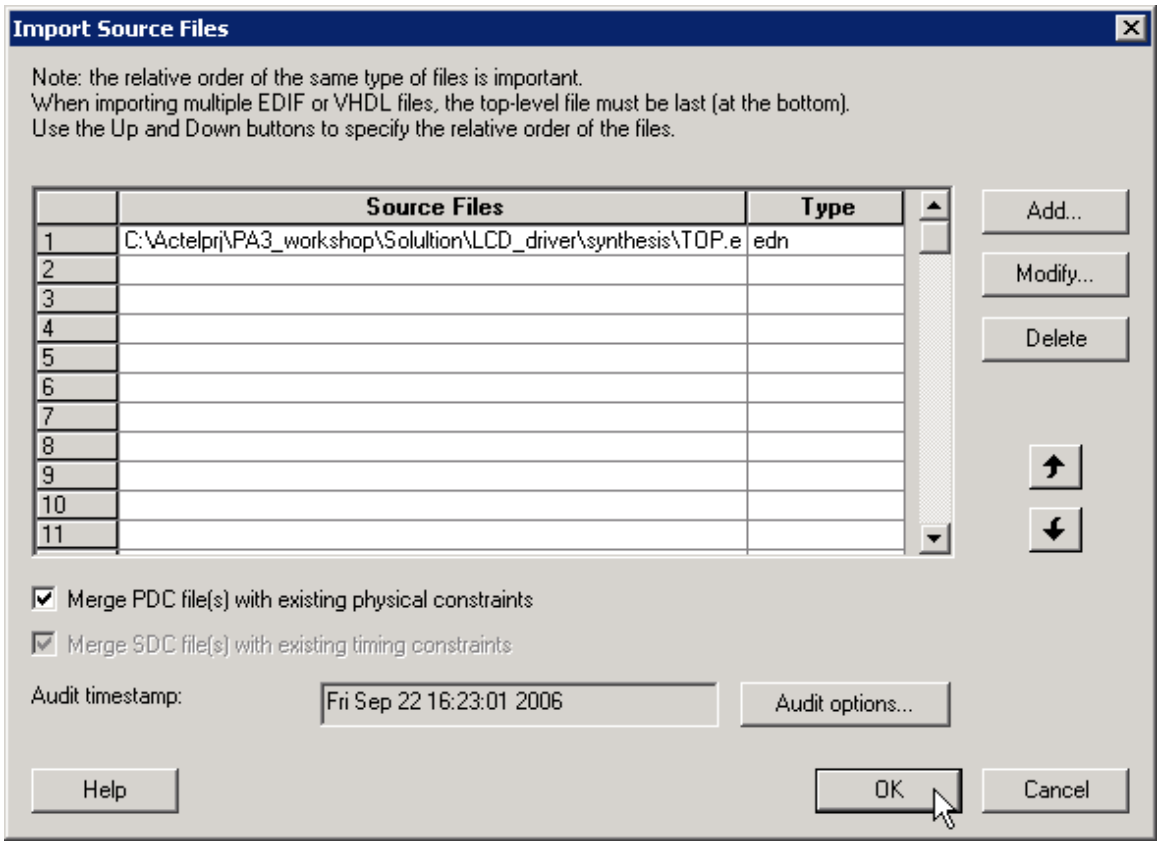


Figure 3 · Import Source Files Dialog Box

2. Click **Add**. The Add Source Files dialog appears.
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. Libero can only audit local files. If you launch Designer from Libero, you need to copy the files locally. Select **Copy locally** to copy these files to your local project folder.

Note: This button is available when design (ADB) opened by Designer is part of a Libero IDEproject.

6. Specifying a priority is useful if you are importing multiple netlist files, GCF files, or PDC, or SDC 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.
7. (IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, and Axcelerator designs only) Select Merge PDC file(s) with existing physical constraints to preserve all existing physical constraints that you have made using ChipPlanner, PinEditor, or the I/O Attribute Editor.
8. (IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, and Axcelerator designs only) Select Merge SDC file(s) with existing timing constraints to preserve all the existing timing constraints already in your design, whether coming

from the Timer tool or a previously imported file. If you import an SDC file and select this option, Designer merges the existing constraints and the constraints contained in the SDC file. In case of a conflict, the new constraint has priority over the existing constraint.

9. To set the audit options for these source files, click **Audit options** and follow the directions in the Audit Options dialog box.
10. 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: Designer may not import file names or paths with spaces. Rename the file or path to remove the spaces, and re-import.

See Also

Auditing Files

`import aux`
`import source`

Importing auxiliary files

Importing Source Files – Copying Files Locally

Designer in Libero IDE cannot import files from outside your project without copying them to your local project folder. You may import source files from other locations, but they are always copied to your local folder. Designer in Libero IDE always audits the local file after you import; it does not audit the original file.

When the Project Manager asks you if you want to copy files "locally", it means 'copy the files to your local project folder'. If you do not wish to copy the files to your local project folder, you cannot import them. Your local project folder contains files related to your Libero IDE project.

Files copied to your local folders are copied directly into their relevant directory: netlists are copied to the *synthesis* folder; source files are copied to *hdl* folder and constraint files to *constraint* folder, etc. The files are also added to the Libero IDE project; they appear in the Files tab.

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, choose **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 open the Modify File Location dialog. The Modify File Location dialog box enables you to specify the correct path so that the design can find the source file(s).
4. Click **Reset to Current Date Time** to associate the file with the current day and time.

Libero IDE and Designer do not audit any CoreConsole generated files. If you regenerate a CoreConsole project you must re-import the configuration file.

See Also

Audit Options dialog box

Auditing Status dialog box

Importing source files

import_source

Importing Auxiliary Files

Auxiliary files are not audited and are treated more as one-time data-entry or data-change events, similar to entering data using one of the interactive editors (e.g. PinEditor or Timer).

If you import the SDC file as an auxiliary, you do not have to re-compile your design. However, auditing is disabled when you import auxiliary files, and Designer cannot detect the changes to your SDC file(s) if you import them as auxiliary files.

Auxiliary Files	File Type Extension	Family
Criticality	*.crt	ACT1, ACT2, ACT3, MX, 1200XL, 3200DX
PIN	*.pin	ACT1, ACT2, ACT3, MX, 1200XL, 3200DX, SX, SX-A, eX
SDC	*.sdc	IGLOO, Fusion, ProASIC3, SX-A, eX, Axcelerator,ProASIC PLUS

Auxiliary Files	File Type Extension	Family
Physical Design Constraint*	*.pdc	IGLOO, Fusion, ProASIC3 and Axcelerator
Value Change Dump	*.vcd	IGLOO, Fusion, ProASIC3, Axcelerator, ProASIC, ProASIC PLUS
Switching Activity Intermediate File/Format	*.saif	IGLOO, Fusion, ProASIC3, Axcelerator, ProASIC, ProASIC PLUS
Design Constraint File	*.dcf	ACT1, ACT2, ACT3, MX, 1200XL, 3200DX, SX, SX-A, eX

Note: Not all PDC commands are supported when a PDC file is imported as an auxiliary file; some must be imported as source files. When importing a PDC file as an auxiliary file, the new or modified PDC constraints are merged with the existing constraints. The software resolves any conflicts between new and existing physical constraints and displays the appropriate message. Most PDC commands can be imported as auxiliary files. PDC commands that are not supported when the PDC file is imported as an auxiliary file are noted in their respective help topics.

Value Change Dump (VCD) files must be imported through SmartPower.

To import an auxiliary file (excluding VCD files):

1. From the **File** menu, choose **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. Some formats (like DCF and SDC) are not allowed to be imported in multiple auxiliary files.

- **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 auxiliary files. Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple 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:

- VCD and SAIF are used by SmartPower for power analysis.
- CRT files are for backwards compatibility with existing designs only.
- File names or paths with spaces may not import into Designer. Rename the file or path, removing the spaces, and re-import.

See Also

Importing source files

import_aux

import_source

Keep existing timing constraints

Keep existing physical constraints

Merge SDC File(s) with Existing Timing Constraints

The Merge SDC file(s) with existing timing constraints checkbox is designed to support an additional "merge or replace" functionality when you import SDC files.

Select **Merge SDC file(s) with existing timing constraints**. to preserve all existing timing constraints that you have made using the Timer GUI or previously imported file. If you import a SDC file and you have this checkbox selected, Designer merges the existing constraints and the constraints existing in the SDC file. In case of a conflict, the new constraint has priority over the existing constraint.

The Merge SDC file(s) with existing timing constraints option is **On** by default. With this option **On**, your timing constraints from the imported SDC files are merged with the existing constraints. When this option is **Off**, all the existing timing constraints are replaced by the constraints in the newly imported SDC files.

See Also

import_aux

import_source

Merge PDC File(s) with Existing Physical Constraints

The Merge PDC file(s) with existing physical constraints option in the Import Source Files dialog box enables you to merge or replace existing constraints when you import new or modified GCF or PDC files.

Select **Merge PDC file(s) with existing physical constraints** to preserve all existing physical constraints that you have entered either using one of the MVN tools (ChipPlanner, PinEditor, or the I/O Attribute Editor) or a previous GCF or PDC file. The software will resolve any conflicts between new and existing physical constraints and display the appropriate message.

The Merge PDC file(s) with existing physical constraints option is Off by default. When this option is Off, all the physical constraints in the newly imported GCF or PDC files are used. All pre-existing constraints are lost. When this option is On, the physical constraints from the newly imported GCF or PDC files are merged with the existing constraints.

See Also

Importing source files

import_source

Design Constraints

Design constraints are usually either requirements or properties in your design. You use constraints to ensure that your design meets its performance goals and pin assignment requirements.

The Designer software supports both timing and physical constraints. In addition, it supports netlist optimization constraints. You can set constraints by either using Actel's interactive tools or by importing constraint files directly into your design session.

Timing Constraints

Timing constraints represent the performance goals for your designs. Designer software uses timing constraints to guide the timing-driven optimization tools in order to meet these goals.

You can set timing constraints either globally or to a specific set of paths in your design.

You can apply timing constraints to:

- Specify the required minimum speed of a clock domain
- Set the input and output port timing information
- Define the maximum delay for a specific path
- Identify paths that are considered false and excluded from the analysis
- Identify paths that require more than one clock cycle to propagate the data
- Provide the external load at a specific port

To get the most effective results from the Designer software, you need to set the timing constraints close to your design goals. Sometimes slightly tightening the timing constraint helps the optimization process to meet the original specifications.

Physical Constraints

Designer software enables you to specify the physical constraints to define the size, shape, utilization, and pin/pad placement of a design. You can specify these constraints based on the utilization, aspect ratio, and dimensions of the die. The pin/pad placement depends on the external physical environment of the design, such as the placement of the device on the board.

There are three types of physical constraints:

- I/O assignments

- Set location, attributes, and technologies for I/O ports

- Specify special assignments, such as VREF pins and I/O banks

- Location and region assignments

- Set the location of Core, RAM, and FIFO macros

- Create Regions for I/O and Core macros as well as modify those regions

- Clock assignments

- Assign nets to clocks

- Assign global clock constraints to global, quadrant, and local clock resources

Note:

Netlist Optimization Constraints

The Designer software enables you to set some advanced design-specific netlist optimizing constraints.

You can apply netlist optimization constraints to:

- Delete or restore a buffer tree
- Manage the fan-outs of the nets
- Manage macro combinations (for example, IO-REG combining)
- Optimize a netlist by removing buffers and/or inverters, propagating constants, and so on

See Also

Constraint Support by Family

Constraint Entry Table

Constraint File Format by Family

Designer Naming Conventions

Constraint Entry

Use the Constraint Entry table to see which tools and file formats you can use to enter constraints for your device family.

Click the name of a constraint, a constraint entry tool, file format type, editor, or checkmark in the table for more information about that item.

Table 1 · Constraint Entry by Tool and File Format

Constraint	SDC	GCF	PDC	PIN	DCF	ChipPlanner	I/O Attribute Editor	PinEditor	Timer/SmartTime	Compile Options
Timing										
Create a clock	X				X				X	
Create a generated clock	X								X	
Set clock latency	X								X	
Set disable timing	X								X	
Set false path	X				X				X	
Set input delay	X								X	
Set load on output port	X				X		X	X	X	
Set maximum delay	X				X				X	
Set minimum delay	X								X	
Set multicycle path	X								X	
Set output delay	X								X	

Constraint	SDC	GCF	PDC	PIN	DCF	ChipPlanner	I/O Attribute Editor	PinEditor	Timer/SmartTime	Compile Options
Physical Placement										
Clocks										
Assign Net to Global Clock		X	X							
Assign Net to Local Clock		X	X			X				
Assign Net to Quadrant Clock			X			X				
Regions										
Assign Macro to Region			X			X				
Assign Net to Region		X	X			X				
Create Region		X	X			X				
Delete Regions			X			X				
Move region			X			X				
Unassign macro(s) driven by net			X			X				
Unassign macro from region			X			X				
I/Os										
Assign I/O to pin		X	X	X		X	X	X		
Assign I/O Macro to Location		X	X			X				

Constraint	SDC	GCF	PDC	PIN	DCF	ChipPlanner	I/O Attribute Editor	PinEditor	Timer/SmartTime	Compile Options
Configure I/O Bank			X			X		X		
Reset attributes on I/O to default settings			X			X	X			
Reset I/O bank to default settings			X			X	X			
Reserve pins			X				X	X		
Unreserve pins			X				X	X		
Unassign I/O macro from location			X			X				
Blocks										
Move Block			X							
Set port block			X			X				
Set Block Options			X							X
Nets										
Assign Net to Global Clock		X	X							
Assign Net to Local Clock		X	X			X				
Assign Net to Quadrant Clock			X			X				
Assign Net to Region		X	X			X				
Reset net's criticality to default level			X							

Constraint	SDC	GCF	PDC	PIN	DCF	ChipPlanner	I/O Attribute Editor	PinEditor	Timer/SmartTime	Compile Options
Set Net's Criticality		X	X							
Unassign macro(s) driven by net			X			X				
Netlist Optimization										
Delete buffer tree		X	X							X
Demote Global Net to Regular Net		X	X							X
Promote regular net to global net		X	X							X
Restore buffer tree		X	X							
Set preserve			X							

See Also

Constraint Support by Family

Constraint File Format by Family

Families Supported

Constraint Support by Family

Use the Constraint Family Support table to see which constraints you can use for your device family. Click the name of a constraint in the table for more information about it.

Table 2 · Constraint Support by Family

	IGLOO	Fusion	ProASIC3	ProASIC PLUS	ProASIC	Axcel-erator	eX	SX-A	SX	MX	DX	ACT1	ACt2/1200 XL	ACT3
Timing														
Create a clock	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Create a generated clock	X	X	X	X	X	X	X	X						
Set clock latency	X	X	X	X	X	X	X	X						
Set disable timing	X	X	X			X (including RTAX-S)							X	X
Set false path	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Set input delay	X	X	X	X		X								
Set load on output port	X	X	X	X	X	X	X	X						
Set maximum delay	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Set minimum delay	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Set multicycle path	X	X	X	X		X								
Set output delay	X	X	X	X		X								
Physical Placement														
Clocks														
Assign Net to Global Clock	X	X	X	X	X									
Assign Net to Local Clock	X	X	X	X	X	X								

	IGLOO	Fusion	ProASIC3	ProASIC PLUS	ProASIC	Axcel-erator	eX	SX-A	SX	MX	DX	ACT1	ACt2/1200 XL	ACT3
Timing														
Assign Net to Quadrant Clock	X	X	X											
Regions														
Assign Macro to Region	X	X	X	X	X	X								
Assign Net to Region	X	X	X	X	X	X								
Create Region	X	X	X	X	X	X								
Delete Regions	X	X	X	X	X	X								
Move Region	X	X	X	X	X	X								
Unassign macro(s) driven by net	X	X	X	X	X	X								
Unassign Macro from Region	X	X	X	X	X	X								
I/Os														
Assign I/O to pin	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Assign I/O Macro to Location	X	X	X	X	X	X								
Configure I/O Bank	X	X	X			X								
Reset attributes on I/O to default settings	X	X	X	X	X	X								
Reset I/O bank to default settings	X	X	X			X								
Reserve pins	X	X	X	X	X	X	X							
Unreserve pins	X	X	X	X	X	X	X							
Unassign I/O macro from location	X	X	X	X	X	X								
Block														
Move Block	X	X	X			X								
Set port block	X	X	X			X								

	IGLOO	Fusion	ProASIC3	ProASIC Plus	ProASIC	Axcel-erator	eX	SX-A	SX	MX	DX	ACT1	ACt2/1200 XL	ACT3
Timing														
Set Block Options	X	X	X			X								
Nets														
Assign Net to Global Clock	X	X	X	X	X									
Assign Net to Local Clock	X	X	X	X	X	X								
Assign Net to Quadrant Clock	X	X	X											
Assign Net to Region	X	X	X	X	X	X								
Reset net's criticality to default level						X								
Set Net's Criticality						X								
Unassign macro(s) driven by net	X	X	X	X		X								
Netlist Optimization														
Delete buffer tree	X	X	X											
Demote Global Net to Regular Net	X	X	X	X	X									
Promote regular net to global net	X	X	X	X	X									
Restore buffer tree	X	X	X	X	X									
Set preserve	X	X	X			X								

See Also

Constraint Entry Table

Constraint File Format by Family

Constraint File Format by Family

Use the File Format by Family table to see which file formats apply to each type of constraint and each device family. Click the name of a file format type in the table for more information about it.

Table 3 · Constraint File Format by Family

Family	Timing		Physical Placement			Netlist Optimiization	
	SDC	DCF	PDC	PIN	GCF	PDC	GCF
IGLOO	X		X				
Fusion	X		X			X	
ProASIC3	X		X				
ProASIC PLUS	X				X		X
ProASIC	X				X		X
Axcelerator	X		X			X	
eX	X	X		X			
SX-A	X	X		X			
SX		X		X			
MX		X		X			
DX		X		X			
ACT3		X		X			
ACT2/1200XL		X		X			
ACT1		X		X			

SDC – Synopsys Design Constraints

PDC – Physical Design Constraints for IGLOO, Fusion, ProASIC3, and Axcelerator

GCF – Design Constraints Format for ProASIC PLUS and ProASIC

DCF – Actel Design Constraints Format

PIN – Pin location constraints

See Also

Constraint Support by Family

Constraint Entry Table

Entering Constraints

You can enter design constraints in the following ways:

- **Importing constraint files**: You can import GCF, PDC, SDC, DCF, or PIN constraint files. The type of file you use depends on which type of device you are designing.

- GCF (ProASIC and ProASIC^{PLUS} families)

- PDC (IGLOO, Fusion, ProASIC3 and Axcelerator families)

- SDC (IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A families)

- DCF (earlier Antifuse families such as eX, SX-A, and SX)

- PIN (only valid for earlier Antifuse families such as eX, SX-A, and SX)

- **Using constraint editor tools**: Designer's constraint editors are graphical user interface (GUI) tools for creating and modifying physical, logical, and timing constraints. Using these tools enables you to enter constraints without having to understand GCF, PDC, or other file syntax. Which constraint editor you use depends on which type of device you are designing.

For **IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator**, use the tools within the MultiView Navigator:

- ChipPlanner - Sets location and region assignments

- PinEditor in MVN - Sets the pin location constraints

- I/O Attribute Editor - Sets I/O attributes

- SmartTime Constraints EditorSmartTime Constraints Editor - Enables you to view and edit timing constraints

For all other families, you will use the following tools:

- ChipEditor - Sets location and region assignments

- PinEditor (non MVN)- Sets I/O attributes and pin location constraints

- Timer - Sets timing constraints

See Also

Constraint Support by Family

Constraint Entry

Constraint File Format by Family

Designer Naming Conventions

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, select **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.

Setting Compile Options

To set compile options

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

Note: Fusion, IGLOO and ProASIC3 Compile options appear by default each time you compile the design. If you have disabled this feature, you can click the Options menu and choose Compile to review/change/reset your Compile options.

Compile options vary according to family.

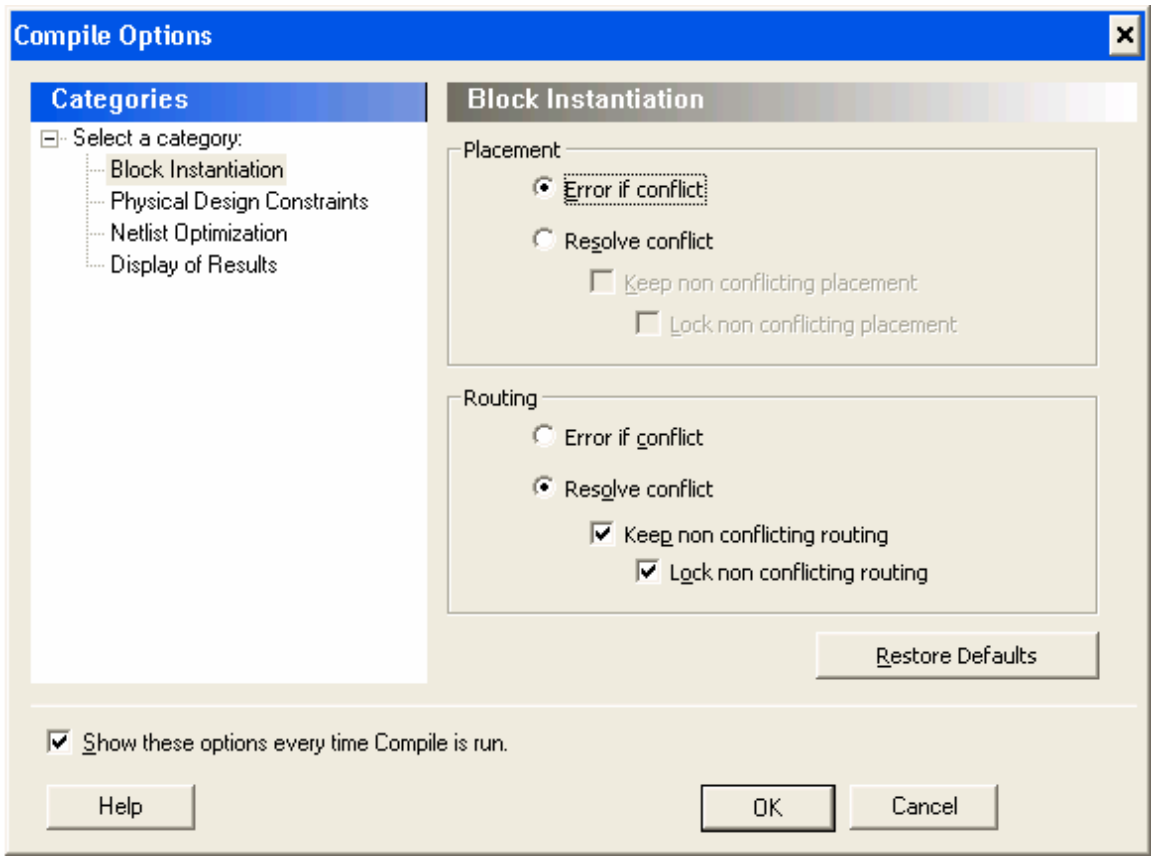
- IGLOO, Fusion, and ProASIC3
- ProASIC, ProASIC^{PLUS}
- Axcelerator
- MX, SX, SX-A, eX

IGLOO, Fusion, and ProASIC3 Compile Options

The IGLOO, Fusion and ProASIC3 Compile Options dialog box enables you to do the following:

- Set your Block Instantiation options (used for conflict resolution when you instantiate multiple blocks)
- Verify Physical Design Constraints
- Perform Globals Management
- Netlist Optimization
- Generate a Compile report in Display of Results
- Set Block Creation options (available only if you are creating a block)

Block Instantiation



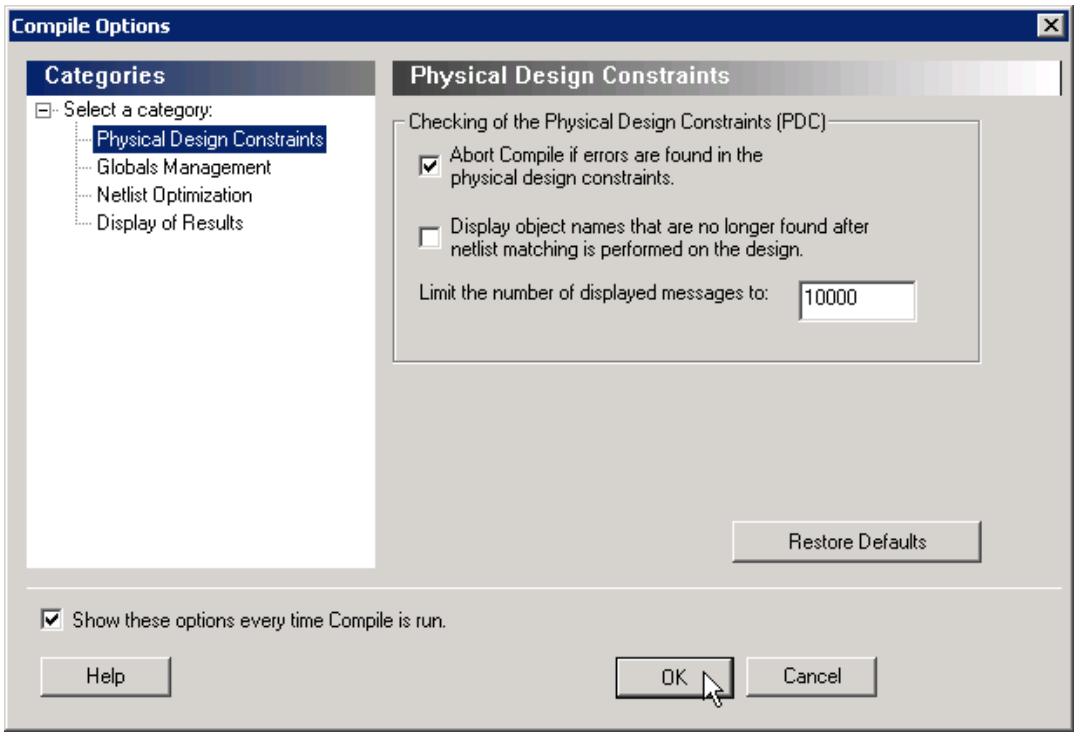
Designer uses the Block Instantiation options to resolve conflicts between multiple blocks in your design. The default options is to return an error if there is overlapping placement between the blocks and resolve any conflict for nets.

This ensures you are aware that the blocks overlap; you can go back and set the placement to resolve the conflicts and it will Compile.

See Conflict resolution in Designer Blocks for more information.

Physical Design Constraints

This interface enables you to verify the Physical Design Constraints (PDC) file.



Checking the Physical Design Constraint (PDC)

Abort Compile if errors are found in the physical design constraints: Changes the "Abort on PDC error" behavior. Select this option to stop the flow if any error is reported in reading your PDC file. If you deselect this option, the tool skips errors in reading your PDC file and just reports them as warnings. The default is ON.

Note: The flow always stops even if this option is deselected in the following two cases:

- If there is a Tcl error (For example, the command does not exist or the syntax of the command is incorrect)
- The assign_local_clock command for assigning nets to LocalClocks fails. This may happen if any floor planning DRC check fails, such as, region resource check, fix macro check (one of the load on the net is outside the local clock region). If such an error occurs, then the Compile command fails. Correct your PDC file to proceed.

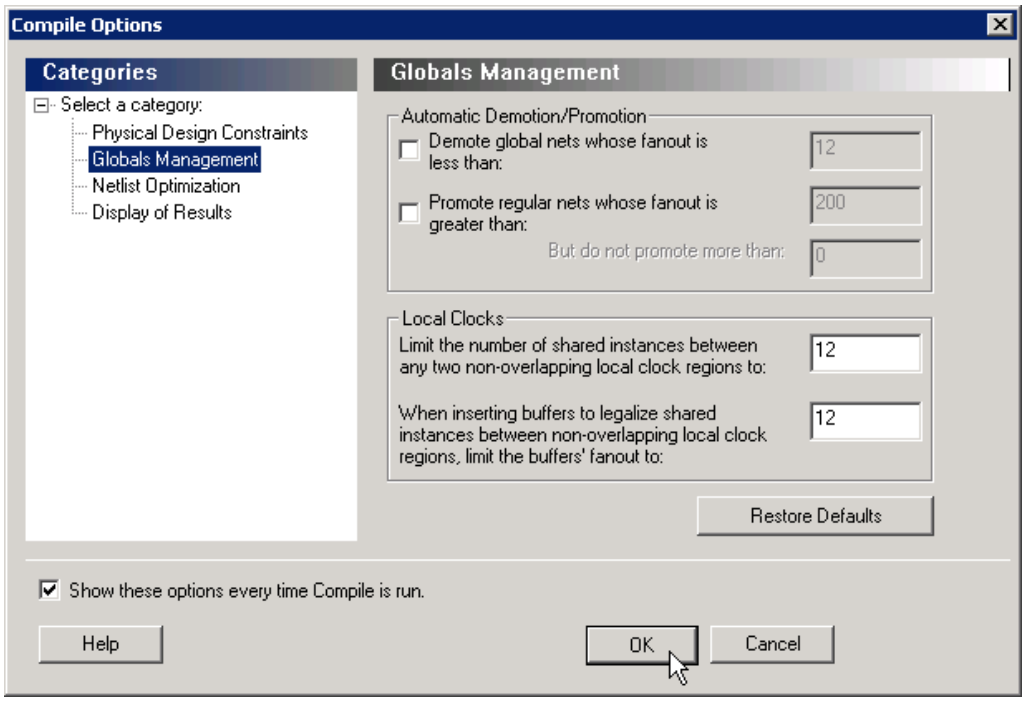
Note: Every time you invoke this dialog box, this option is reset to its default value ON. This is to ensure that you have a correct PDC file.

Display object names that are no longer found after netlist matching is performed on the design: Displays netlist objects in the PDC that are not found in the imported netlist during the Compile ECO mode. Select this option to report netlist objects not found in the current netlist when reading the internal ECO PDC constraints. The default is OFF.

Limit the number of displayed messages to: Defines the maximum number of errors/warnings to be displayed in the case of reading ECO constraints. The default is 10000 messages.

Globals Management

The interface provides a global control to the Compile component of the design flow.



Automatic Demotion/Promotion

Demote global nets whose fanout is less than: Enables the global clock demotion of global nets to regular nets. By default, this option is OFF. The maximum fanout of a demoted net is 12.

Note: A global net is not automatically demoted (assuming the option is selected) if the resulting fanout of the demoted net is greater than the max fanout value. Actel recommends that the automatic global demotion only act on small fanout nets. Actel recommends that you drive high fanout nets with a clock network in the design to improve timing and routability.

Promote regular nets whose fanout is greater than: Enables global clock promotion of nets to global clock network. By default, this option is OFF. The minimum fanout of a promoted net is 200.

But do not promote more than: Defines the maximum number of nets to be automatically promoted to global. The default value is 0. This is not the total number as nets need to satisfy the minimum fanout constraint to be promoted. The promote_globals_max_limit value does not include globals that may have come from either the netlist or PDC file (quadrant clock assignment or global promotion).

Note: Demotion of globals through PDC or Compile is done before automatic global promotion is done. You may exceed the number of globals present in the device if you have nets already assigned to globals or quadrants from the netlist or by using a PDC file. The automatic global promotion adds globals on what already exists in the design.

Local Clocks

Limit the number of shared instances between any two non-overlapping local clock regions to: Defines the maximum number of shared instances allowed to perform the legalization. It is also for quadrant clocks.

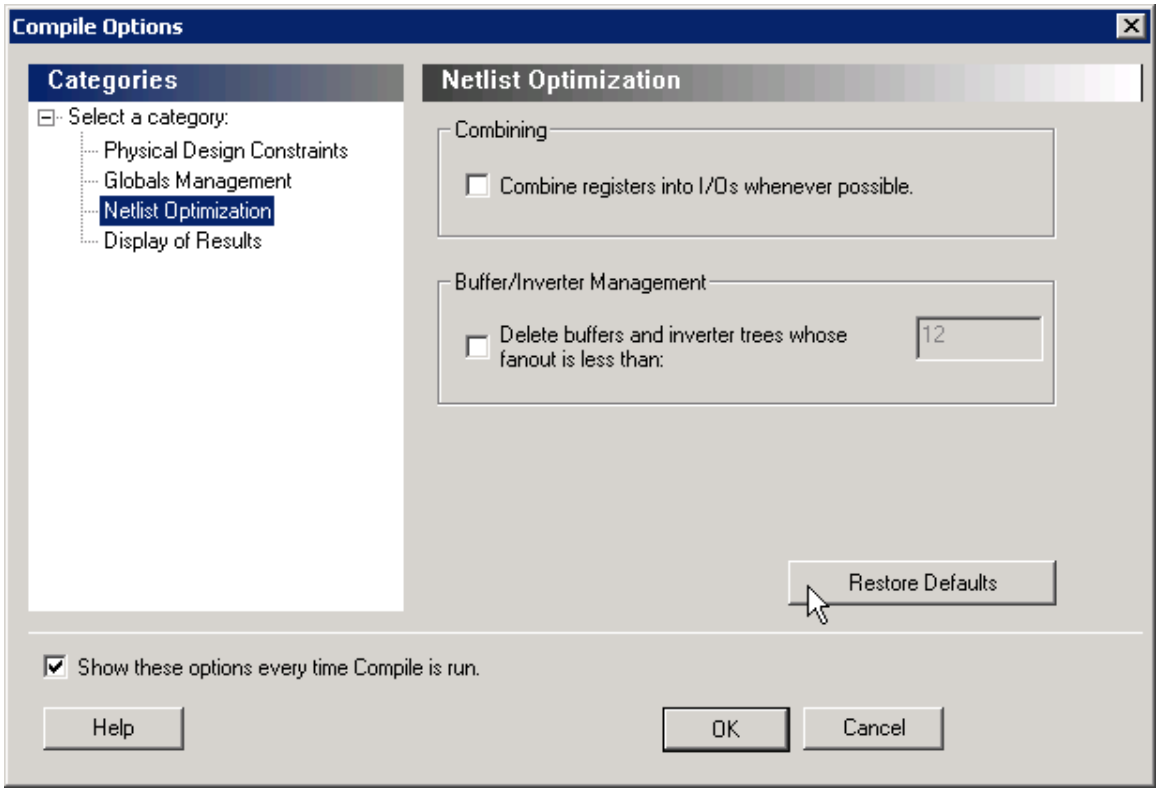
The maximum number of instances allowed to be shared by 2 local clock nets assigned to disjoint regions to perform the legalization (default is 12, range is 0-1000). If the number of shared instances is set to 0, no legalization is performed.

When inserting buffers to legalize shared instances between non-overlapping local clock regions, limit the buffers' fanout to: Defines the maximum fanout value used during buffer insertion for clock legalization. Set the value to 0 to disable this option and prevent legalization (default value is 12, range is 0-1000). If the value is set to 0, no buffer insertion is performed. If the value is set to 1, there will be one buffer inserted per pin.

Note: If you assign quadrant clock to nets using MultiView Navigator, no legalization is performed.

Netlist Optimization

This interface allows you to perform netlist optimization.



Combining

Combine registers into I/O wherever possible: Combines registers at the I/O into I/O-Registers. Select this option for optimization to take effect. By default, this option is OFF.

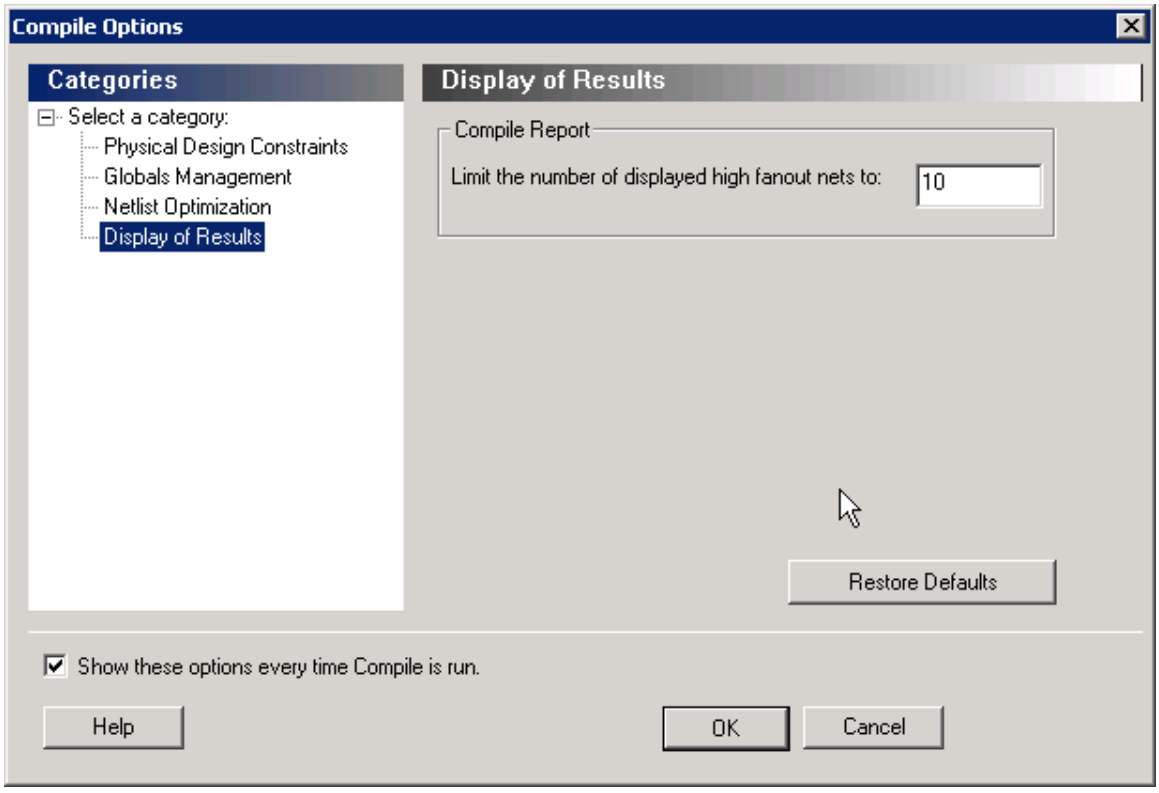
Buffer/Inverter Management

Delete buffers and inverter trees whose fanout is less than: Enables buffer tree deletion on the global signals from the netlist. The buffer and inverter are deleted. By default, this option is OFF. The maximum fanout of a net after buffer tree deletion is 12.

Note: A net does not automatically remove its buffer tree (assuming the option is on) if the resulting fanout of the net (if the buffer tree was removed) is greater than the max fanout value. Actel recomends that the automatic buffer tree deletion should only act on small fanout nets. From a routability and timing point of view, it is not recommended to have high fanout nets not driven by a clock network in the design.

Display of Results

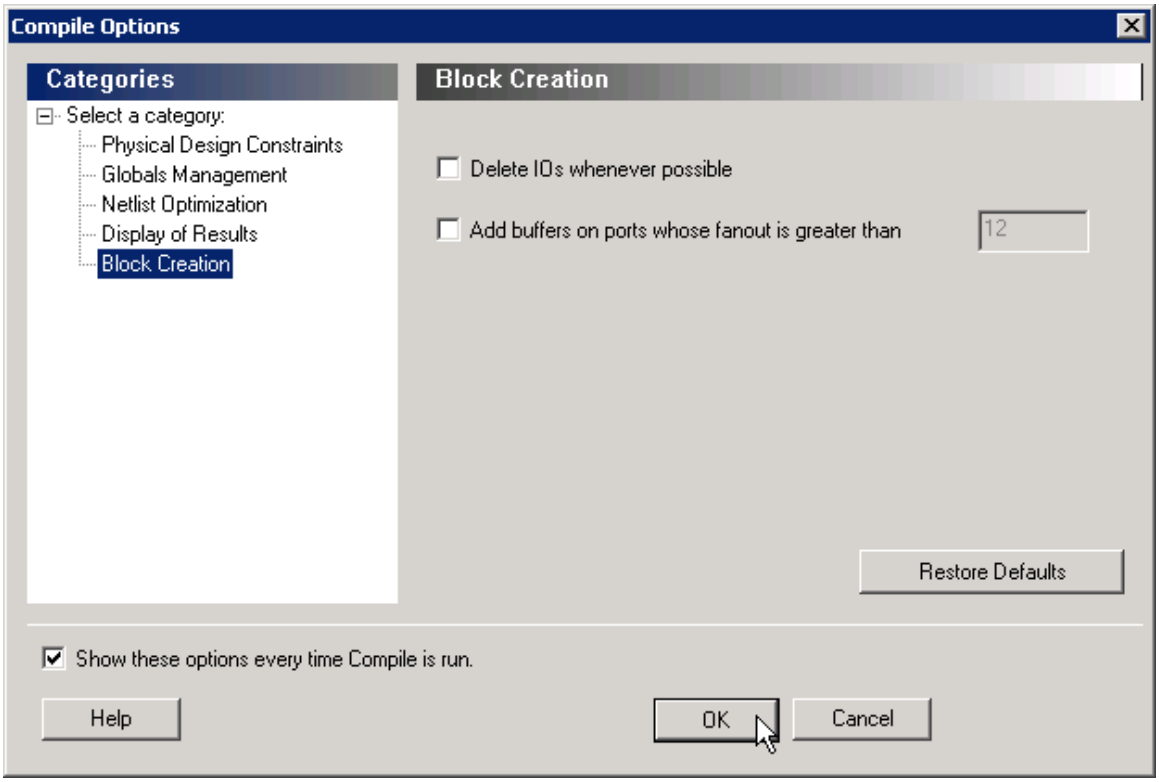
This interface lets you generate a Compile report.



Compile Report

Limit the number of displayed high fanout nets to: Enables flip-flop net sections in the compile report and defines the number of nets to be displayed in the high fanout. The default value is 10.

Block Creation (available only when creating Designer Blocks)



Delete I/Os whenever possible - Deletes I/Os in the block during compile (except TRIBUFF and BIBUFF, because they cannot be removed). Useful if you have I/Os in your design but want to create a block anyway.

Add buffers on ports whose fanout is greater than <value> - Adds buffers on ports with a fanout greater than a value you specify. This option enables more predictable block timing. For example, if you have a net with a fanout of 100 the net will be unrouted. If you add a buffer, the output of the buffer is routed and the routing is preserved.

See Also

compile

ProASIC and ProASIC[PLUS] Compile Options

Include RAM and I/O in Spine and Net Regions

This option affects the behavior of the following:

- The use_global constraint
- The set_net_region constraint
- The creation of spines in the MultiView Navigator

Selecting **Include RAM and I/O in Spine and Net Regions** enables you to assign memory and I/O to spine (LocalClock) and net regions.

When this option is selected, Designer applies the use_global and set_net_region constraints to core cells, memory, and I/O. When cleared, Designer applies the use_global and set_net_region constraints to core cells only. For new designs, this box is automatically checked. For designs created with v5.1 or earlier, this option is cleared by default. If you change this default setting, you must recompile your design.

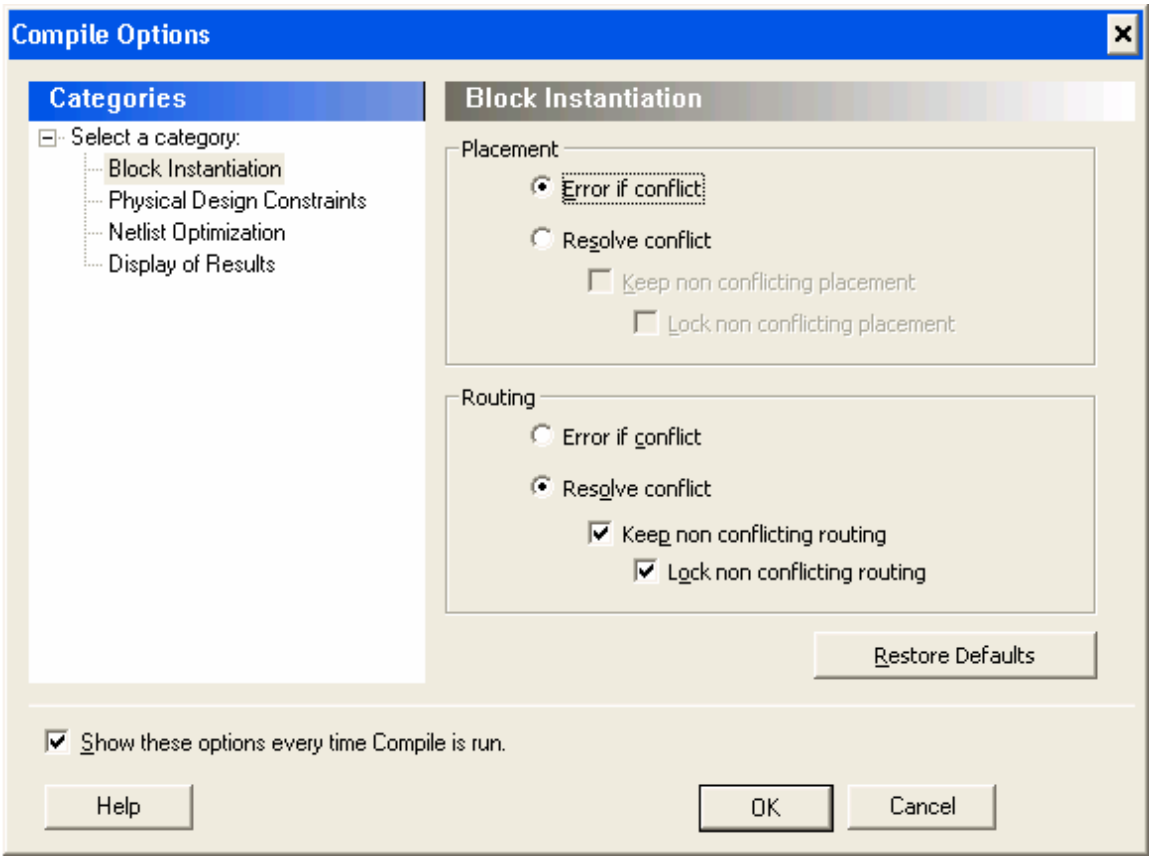
This option also determines whether memory and I/O are included in a LocalClock region that you create with the ChipPlanner tool. If selected, memory and I/O are included. If cleared, they are excluded.

Axcelerator Compile Options

The Axcelerator Compile Options dialog box enables you to do the following:

- Set your Block Instantiation options (used for conflict resolution when you instantiate multiple blocks)
- Verify Physical Design Constraints
- Netlist Optimization
- Generate a Compile report in Display of Results
- Set Block Creation options (available only if you are creating a block)

Block Instantiation



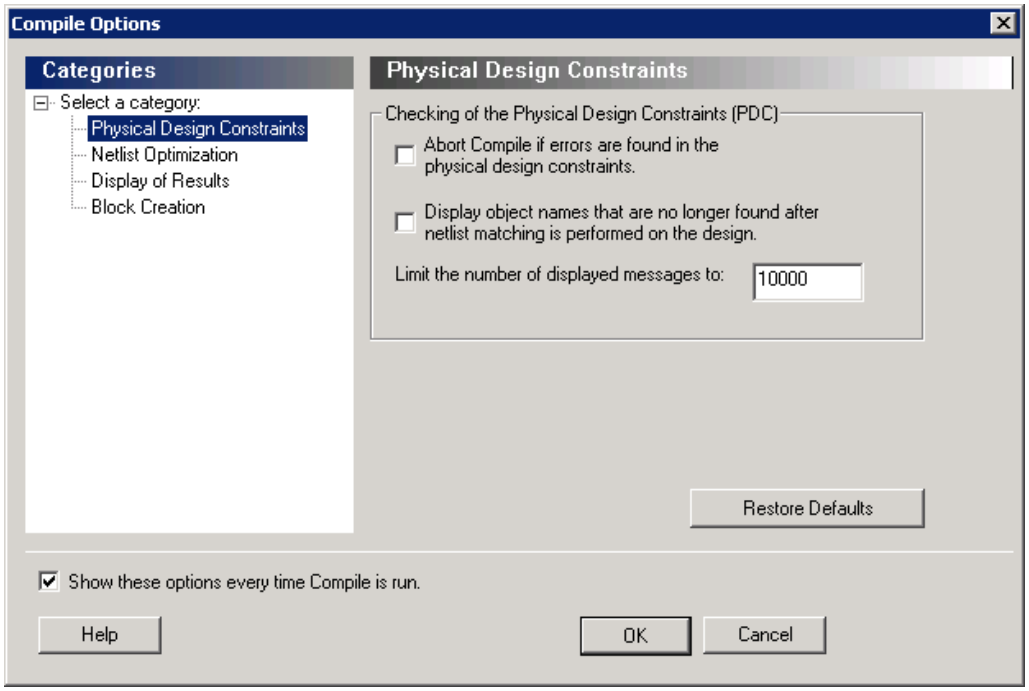
Designer uses the Block Instantiation options to resolve conflicts between multiple blocks in your design. The default options is to return an error if there is overlapping placement between the blocks and resolve any conflict for nets.

This ensures you are aware that the blocks overlap; you can go back and set the placement to resolve the conflicts and it will Compile.

See Conflict resolution in Designer Blocks for more information.

Physical Design Constraints

This interface enables you to verify the Physical Design Constraints (PDC) file.



Checking the Physical Design Constraint (PDC)

Abort Compile if errors are found in the physical design constraints: Changes the "Abort on PDC error" behavior. Select this option to stop the flow if any error is reported in reading your PDC file. If you deselect this option, the tool skips errors in reading your PDC file and just reports them as warnings. The default is ON.

Note: The flow always stops even if this option is deselected in the following two cases:

- If there is a Tcl error (For example, the command does not exist or the syntax of the command is incorrect)
- The assign_local_clock command for assigning nets to LocalClocks fails. This may happen if any floor planning DRC check fails, such as, region resource check, fix macro check (one of the load on the net is outside the local clock region). If such an error occurs, then the Compile command fails. Correct your PDC file to proceed.

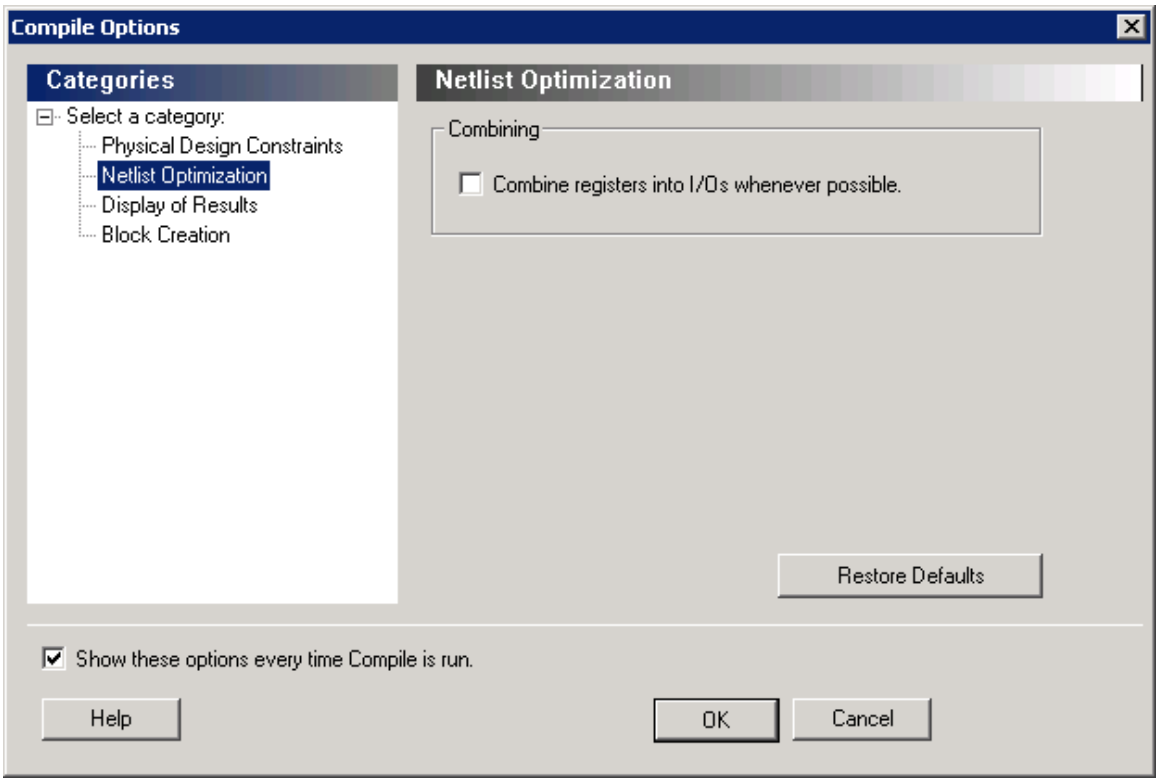
Note: Every time you invoke this dialog box, this option is reset to its default value ON. This is to ensure that you have a valid PDC file.

Display object names that are no longer found after netlist matching is performed on the design: Displays netlist objects in the PDC that are not found in the imported netlist during the Compile ECO mode. Select this option to report netlist objects not found in the current netlist when reading the internal ECO PDC constraints. The default is OFF.

Limit the number of displayed messages to: Defines the maximum number of errors/warnings to be displayed in the case of reading ECO constraints. The default is 10000 messages.

Netlist Optimization

This interface allows you to perform netlist optimization.



Combining

Combine registers into I/O wherever possible: Combines registers at the I/O into I/O-Registers. Select this option for optimization to take effect. By default, this option is OFF.

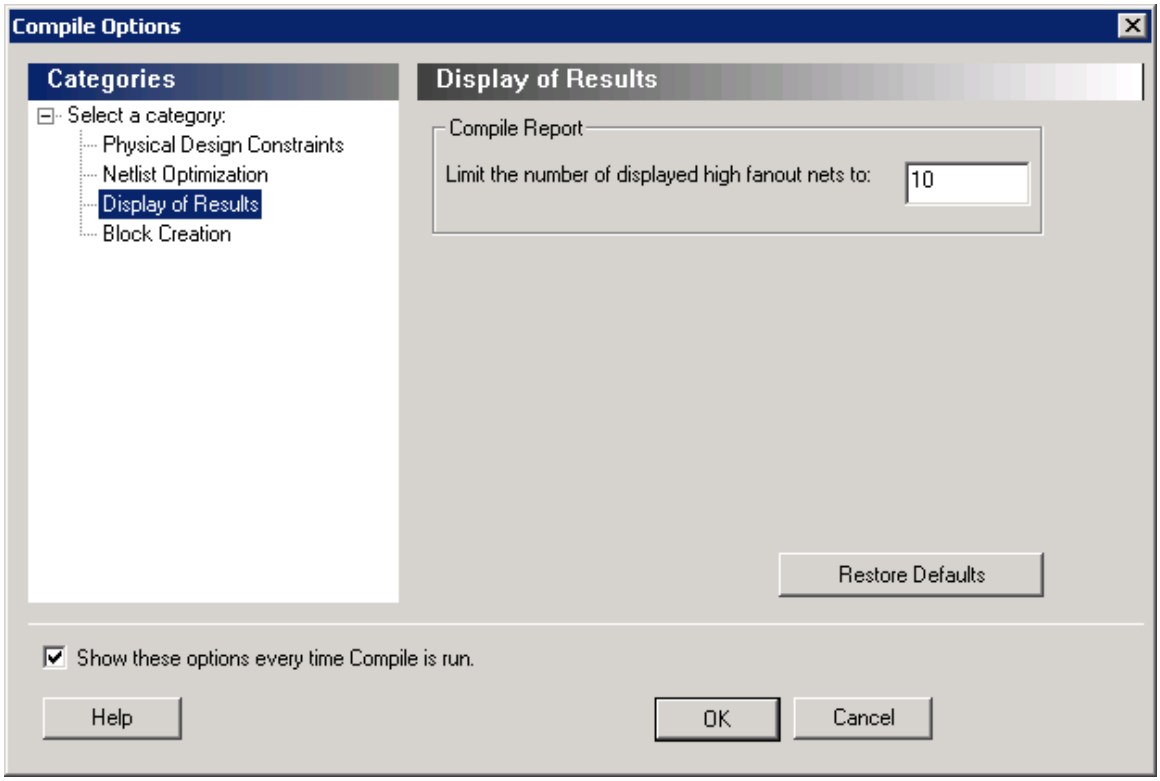
Buffer/Inverter Management

Delete buffers and inverter trees whose fanout is less than: Enables buffer tree deletion on the global signals from the netlist. The buffer and inverter are deleted. By default, this option is OFF. The maximum fanout of a net after buffer tree deletion is 12.

Note: A net does not automatically remove its buffer tree (assuming the option is on) if the resulting fanout of the net (if the buffer tree was removed) is greater than the max fanout value. Actel recomends that the automatic buffer tree deletion should only act on small fanout nets. From a routability and timing point of view, it is not recommended to have high fanout nets not driven by a clock network in the design.

Display of Results

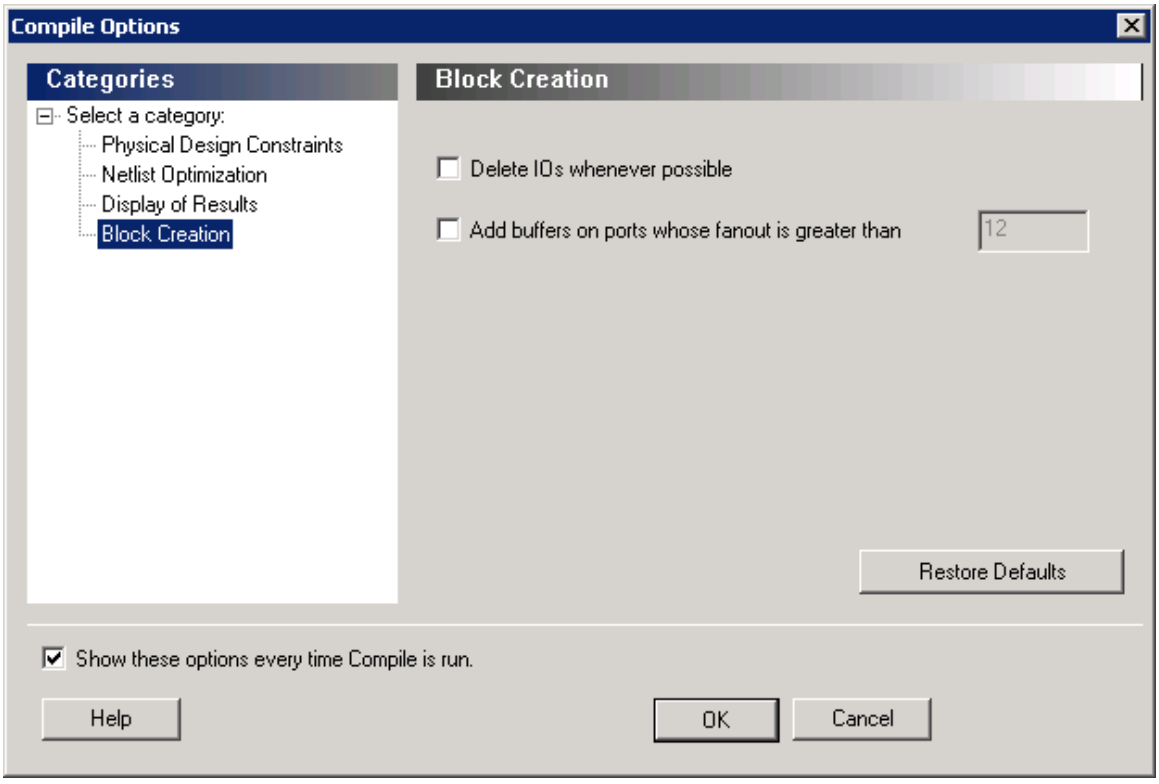
This interface lets you generate a Compile report.



Compile Report

Limit the number of displayed high fanout nets to: Enables flip-flop net sections in the compile report and defines the number of nets to be displayed in the high fanout. The default value is 10.

Block Creation (available only when creating Designer Blocks)



Delete I/Os whenever possible - Deletes I/Os in the block during compile (except TRIBUFF and BIBUFF, because they cannot be removed). Useful if you have I/O's in your design but want to create a block anyway.

Add buffers on ports whose fanout is greater than <value> - Adds buffers on ports with a fanout greater than a value you specify. This option enables more predictable block timing. For example, if you have a net with a fanout of 100 the net will be unrouted. If you add a buffer, the output of the buffer is routed and the routing is preserved.

See Also

compile

MX, SX, SX-A, eX Compile Options

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 check box.

If this option is **Off**, or cleared, 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 selected, 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**.

Fanout Messages

Use the control slider in the Messages area to control the warning level. Use the control slider to specify the fanout limit that the Compile step checks against. Setting the control slider to '0' informs the system to use the system defaults. Any non-zero value replaces the system default value for the fanout limit with the user-specified value. Typically, this value range is 1 to 24.

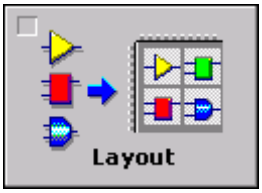
This does not adjust the fanout of the design and it has no effect on the netlist. This only adjusts the warning level by controlling what level of fanout checking you want to be warned about during Compile. Changing this fanout limit option does not invalidate the Compile design state.

Running Layout

Use Layout to place and route your design.

To run Layout:

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



2. **Layout Options**. Select your Layout options and Click **OK**. Layout options are family specific:
    - IGLOO, Fusion, and ProASIC3 Layout Options
    - ProASICPLUS and ProASIC Layout Options
    - Axcelerator Layout Options
    - eX, SX, and SX-A Layout Options
    - ACT, MX, and DX Layout Options

IGLOO, Fusion, and ProASIC3 Layout Options

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

The I/O Bank Assigner and Global Planner run automatically after you click **OK** in the **Layout Options** dialog box. The I/O Bank Assigner automatically assigns technologies to all I/O banks that have not been assigned a technology. The Global Planner automatically assigns global nets to clock conditioning circuit (CCC) locations on the chip in the design.

Note: All I/O technologies assigned to I/O banks by the I/O Bank Assigner in Layout are unlocked.

Timing-Driven

Select this option to run Timing-Driven Layout. The primary goal of timing-driven layout is to meet timing constraints, specified by you or generated automatically. Timing-driven Layout typically delivers better performance than Standard layout.

If you do not select Timing-driven layout, Designer runs Standard layout. Standard layout targets efficient usage of the chip resources. Chip performance is not optimized. Timing constraints are not considered by the Layout in standard mode, although 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.

If your design has multiple scenarios, you can select a scenario from the pull-down list to perform timing driven layout.

Power-Driven

Select this option to run Power-Driven Layout. The primary goal of power-driven layout is to reduce dynamic power while still maintaining timing constraints.

To get the most out of Power-Driven Layout, it is recommended to:

1. Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout meet the constraints that are needed).
3. Perform Layout with **Timing-Driven**, **Run Place**, and **Run Route** options checked.
4. Resolve worst case setup and maximum delay violations.
5. Generate an SDF back-annotation file.
6. Perform a post layout back-annotated simulation using this SDF file, and export a VCD (Value Change Dump) file that will capture real activities for each net.
7. Import this VCD file in Designer using the **Import Auxiliary** option from the **File** menu.
8. Perform Layout with **Timing-Driven** and **Power-Driven** checked. Run Place and Route.
9. Verify that your timing constraints are still met with SmartTime.
10. Analyze your power with SmartPower.

In case you do not have simulation vectors for your design, the following alternative flow is recommended:

1. Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout to meet the constraints that are needed).
3. Perform Layout with **Timing-Driven**, **Run Place**, and **Run Route** options checked.
4. Resolve worst case setup and maximum delay violations.
5. Verify that your timing constraints are still met with SmartTime.
6. Open SmartPower and set clock frequencies and toggle rates for the different clocks. Clock frequencies can be imported from your timing constraints. Refer to Initialize Frequencies for more information.
7. Perform Layout with **Timing-Driven**, and **Power-Driven** options checked. Run Place and Route.
8. Verify that your timing constraints are still met with SmartTime.
9. Analyze your power with SmartPower

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 selected by default. If your design has already been placed but not routed, this box is cleared 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.

Incremental options apply to the entire design. For more detailed control of the placer behavior (such as, to fix placement of a portion of the design), use the MultiView Navigator tools or set fixed attributes on the placed instances via PDC constraint files.

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 check box is cleared.

- **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 ECO. (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, deactivate timing-driven mode, or select a bigger device and rerun Layout. Also, see the Advanced Layout options for your device.

There is no "Fix" option for the router. In incremental mode the router tries to preserve the existing routing; there is no guarantee that it will be preserved. Therefore the timing characteristics of the previously routed portion of the design may change, even if the placement was fixed for that portion of the design. The chance of this is quite small, and the router will print the list of nets that have fixed terminals (i.e. those nets whose every pin's macro has the placement FIX attribute).

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.

Click the Advanced button to set Timing-Driven options.

See Also

Running Layout

IGLOO, Fusion, and ProASIC3 Advanced Layout Options

To set these advanced options during Layout, click **Advanced** in the Layout dialog box. The Advanced Layout options are only available in timing-driven Layout mode.

High Effort Layout Mode

This option turns on netlist optimizations to obtain better performance. Layout runtime will increase when this option is selected. You can also combine this option with the Multi-Pass mode to achieve the best possible performance.

In the regular flow the compile step in Designer would modify the netlist to make use of efficient resources on the chip, such as global networks and special macros. When the **High Effort Layout** option is turned on, the placer could further change the mapping of the logic components, preserving the original functionality of the design. The changed netlist is then used in all post-layout Designer tools including back-annotation.

The names and types of the combinational core logic primitives may change. All other logic cells (such as registers, memory, I/Os or clocks) or combinational logic primitives that are assigned a physical constraint (locked at a location, assigned to a region, or part of a block component), referred in a timing constraint, or have a preserve property, will remain unchanged.

When the **Lock Existing Placement** option is also turned on, the placer runs in regular effort mode.

Note: If you change the High Effort Setting you must re-run the placer and router to complete Layout.

Sequential Optimization

This option turns on optimization of sequential cells in the High Effort Layout mode. This typically enables register retiming without disturbing timing latency. The names of registers may change unless they are assigned a physical constraint (locked at a location, assigned to a region, or part of a block component), referred in a timing constraint, or have a preserve property. Other restrictions may also apply.

Router

Repair Minimum Delay Violations

With this option selected, layout will perform an additional route that will attempt to repair paths that have minimum delay and hold time violations. This is done by increasing the length of routing paths and inserting routing buffers to add delay to paths. Since placement will remain unchanged and no additional tiles or modules will be inserted, the amount of delay inserted is limited. As a result, this function is best suited to repair paths with small (0 to 3 ns) hold and minimum delay violations. Paths with large violations will likely improve, but for a complete repair of these paths, manual placement or source code modification may be necessary. Every effort will be made to avoid creating max-delay timing violations on worst case paths.

To get the most out of repair minimum delay violations, it is recommended to:

1. Enter max-delay, min-delay, setup and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout to meet the constraints that are needed).
3. Perform Layout with **Timing Driven**, **Run Place**, **Run Route** and optionally **Run incrementally** enabled.
4. Resolve worst case setup and max-delay violations before running minimum delay violations repair.
5. After worst case max-delay timing is resolved, evaluate timing in SmartTime's Timing Analyzer in minimum delay analysis mode to check for hold time and minimum delay violations.
6. Run repair minimum delay violations with incremental route enabled.
   The repair minimum delay violations tool will attempt to fix all hold time and minimum delay violations by lengthening routing delay paths and inserting routing buffers. As delay is added to paths, worst case max-delay timing is verified to avoid creating new max-delay timing violations. Designer will report the worst minimum slack and the number of violating paths in the log window. In some cases, additional improvement can occur by running repair minimum delay violations multiple times with **Run Incrementally** enabled.
7. Perform both maximum and minimum delay timing analysis to check the timing. Manual placement or source code modification may be necessary to repair all minimum delay violations.
8. After making placement or source code changes, run incremental route and repair minimum delay violations, and then analyze timing again.

Additional Factors

Runtime may vary greatly with the number of paths that need repair, the number of nets in those paths, and the resources available for the tool to insert delay. Over-constraining paths will increase runtime, but will not likely improve results .

The tool will only work on paths that have min delay and hold time constraints. However, other paths that share common nets to the constrained paths may be inadvertently affected.

It is recommended to run minimum delay violations repair with incremental route. This will ensure that paths which do not have minimum delay violations are preserved.

Repair will be performed on:

- Register to register paths where both registers are on the same global or non-global clock
- Register to register paths where the registers are on different clock networks and a minimum delay constraint exists
- Input to register, register to output, clock to out, input to output paths with minimum delay or hold constraint.

You may select programmable input delays to increase delay on input to register paths for devices that support the feature.

Restore Defaults

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

ProASIC^PLUS and ProASIC Advanced Layout Options

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

Layout Timing Weight

Setting this option to values within a recommended range of 1-4 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.

Value 4 (default) puts maximum emphasis on the performance; value 1 shifts priority to routability, but still evaluates timing goals during layout.

In general, the performance of a non-congested design increases with a higher timing weight setting. In highly congested designs the performance improves with lower timing weight settings because of better routability. Use intermediate values to balance the performance-routability tradeoff for your design.

For designs that are very congested, you may want to put even more emphasis to the routability than level 1 provides. In this case try running standard-mode placer, followed by the timing-driven router. If this is insufficient, run both place-and-route in standard mode.

Note: If you change the Timing Weight you must re-run the placer to complete routing. Changing the Timing Weight has no effect if you do not re-run the placer.

Restore Defaults

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

Axcelerator Layout Options

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

The I/O Bank Assigner runs automatically after you click OK in the Layout Options dialog box. The I/O Bank Assigner automatically assigns technologies to all I/O banks that have not been assigned a technology.

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 delivers performance that is superior to Standard Layout; Timing-Driven Layout is selected by default.

Layout is run in Standard mode when the Timing-Driven check box is cleared. The goal of Standard Layout is to optimize for routability.

Power-Driven

Select this option to run Power-Driven Layout. The primary goal of power-driven layout is to reduce dynamic power while still maintaining timing constraints.

To get the most out of Power-Driven Layout, it is recommended to:

1. Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout meet the constraints that are needed).
3. Perform Layout with **Timing-Driven** and **Power-Driven** options checked. Run Place and Route.
4. Verify that your timing constraints are still met with SmartTime.
5. Analyze your power with SmartPower.

Run Place

Select this option to run the placer during Layout. If you have not run Layout before, Run Place is selected by default. If your design has already been placed but not routed, this check box is not selected. 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 lock previous placement data for the next incremental placement run.

Effort Level

Use the Effort Level slider to increase the placement effort. 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. If you have not run Layout before, Run Route is selected. Run Route is also selected if your previous Layout run completed with routing failures.

Incremental routing is available for Axcelerator devices. When activated, the option sets the previous routing information as the initial starting point. To use the incremental routing option in the script mode, see the Advanced Tcl Layout options for Axcelerator (in the Tcl Scripting section).

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 select both *Run Place* and *Run Route*.

Axcelerator Advanced Layout Options

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

Router

Repair Minimum Delay Violations

With this option selected, layout will perform an additional route that will attempt to repair paths that have minimum delay and hold time violations. This is done by increasing the length of routing paths and inserting routing buffers to add delay to paths. Since placement will remain unchanged and no additional tiles or modules will be inserted, the amount of delay inserted is limited. As a result, this function is best suited to repair paths with small (0 to 3 ns) hold and minimum delay violations. Paths with large violations will likely improve, but for a complete repair of these paths, manual placement or source code modification may be necessary. Every effort will be made to avoid creating max-delay timing violations on worst case paths.

To get the most out of repair minimum delay violations, it is recommended to:

1. Enter max-delay, min-delay, setup and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout to meet the constraints that are needed).
3. Perform Layout with **Timing Driven**, **Run Place**, **Run Route** and/or **Run incrementally** enabled.
4. Resolve worst case setup and max-delay violations before running minimum delay violations repair.
5. After worst case max-delay timing is resolved, evaluate timing in SmartTime's Timing Analyzer in minimum delay analysis mode to check for hold time and minimum delay violations.
6. Run repair minimum delay violations with incremental route enabled.
   The repair minimum delay violations tool will attempt to fix all hold time and minimum delay violations by lengthening routing delay paths and inserting routing buffers. As delay is added to paths, worst case max-delay timing is verified to avoid creating new max-delay timing violations. Designer will report the worst minimum slack and the number of violating paths in the log window. In some cases, additional improvement can occur by running repair minimum delay violations multiple times with **Run Incrementally** enabled.
7. Perform both maximum and minimum delay timing analysis to check the timing. Manual placement or source code modification may be necessary to repair all minimum delay violations.
8. After making placement or source code changes, run incremental route and repair minimum delay violations, and then analyze timing again.

Note: Runtime may vary greatly with the number of paths that need repair, the number of nets in those paths, and the resources available for the tool to insert delay. Over-constraining paths will increase runtime, but will not likely improve results.

The tool will only work on paths that have min delay and hold time constraints. However, other paths that share common nets to the constrained paths may be inadvertently affected.

Actel recommends that you run minimum delay violations repair with incremental route. This will ensure that paths which do not have minimum delay violations are preserved.

Repair will be performed on:

- Register to register paths where both registers are on the same global or non-global clock
- Register to register paths where the registers are on different clock networks and a minimum delay constraint exists
- Input to register, register to output, clock to out, input to output paths with minimum delay or hold constraint.

You may select programmable input delays to increase delay on input to register paths for devices that support the feature.

Restore Default

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

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.

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 five 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 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, Designer runs standard layout. 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.

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 five times as long as a normal layout.

Restore Default

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

See Also

Running Layout

ACT, MX, and DX Layout 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 the **Lock Existing Placement** option is selected in the Layout dialog box, all unchanged macros are treated as locked (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 ChipEditorfor details on locking macros.

ProASIC and ProASIC^{PLUS} Placement Constraint File (GCF)

For ProASIC and ProASIC^{PLUS} designs, you can export a GCF constraint file to get all of the constraint information. From the **File** menu, choose **Export > Constraint Files**, type a file name and click **Save**, and then select **All GCF constraints** in the **Export GCF File** dialog box. Blocks with locked placement constraints generate locked placement constraints, while the others generate initial placement constraints. You can edit a GCF file to remove existing constraints or add new constraints. You must then import the modified GCF file as well as the netlist back into Designer. See Importing Source Files for more information about importing files.

Running 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 specified criteria.

Note:

- Before running Multiple Pass Layout, you need to save your design.
- Multiple Pass Layout is supported in the following families: IGLOO, Fusion, ProASIC3, Axcelerator, ProASIC^{PLUS}, ProASIC, SX-A, and eX.
- Multiple Pass Layout saves your design file with the pass that has the best layout results. 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, select **Save As**.
- Four types of reports (timing, maximum delay timing violations, minimum delay timing violations, and power) for each pass will be written out to the working directory to assist you in later analysis:
- *<adbFileName>_timing_r<runNum>_s<seedIndex>.rpt*
- *<adbFileName>_timing_violations_r<runNum>_s<seedIndex>.rpt*
- *<adbFileName>_timing_violations_min_r<runNum>_s<seedIndex>.rpt*
- *<adbFileName>_power_r<runNum>_s<seedIndex>.rpt*
- *<adbFileName>_iteration_summary.rpt* provides additional details about the saved files

To configure your multiple pass options:

1. When running Layout, select **Use Multiple Passes** in the Layout Options dialog box.

2. Click **Configure**. The Multi-Pass Configuration dialog box appears.
3. Set the options and click **OK**.

Number ofpasses: Set the number of passes (iterations) using the slider. 1 is the minimum and 25 is the maximum. The recommended number of passes is 5.

Start at seed index: Set the specific index into the array of random seeds which is to be the starting point for the passes.

Measurement: Select the measurement criteria you want to compare layout results against.

- **Slowest clock**: Select to use the slowest clock frequency in the design in a given pass as the performance reference for the layout pass.
- **Specific clock**: Select to use a specific clock frequency 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 SmartTime or SDC.
- **Maximum delay**: Select to examine timing violations (slacks) obtained from maximum delay analysis.
- **Minimum delay**: Select to examine timing violations (slacks) obtained from minimum delay analysis.
- **Select by**: Worst Slack or Total Negative Slack to specify the slack criteria.
    - When Worst Slack is selected, the most amount of negative slack (or least amount of positive slack if all constraints are met) for each pass is identified, and then the largest value out of all passes determines the best pass.
    - When Total Negative Slack is selected, the sum of negative slacks from the first 100 paths for each pass is identified, and then the largest value out of all the passes determines the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass.
- **Stop on first path without violations**: Select to stop performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).
- **Total power:** Select to determine the best pass to be the one that has the lowest total power (static + dynamic) out of all layout passes.

Save design file for each pass: Select to save the design *.adb file for each pass. By default, only the best result is saved to your design. With this option, every pass stores its design file as *<adbFileName>_r<runNum>_s<seedIndex>.adb*. The "best"-pass design will also be written back to the original *.adb file. Saving all results does take more disk space, but allows you to analyze the result of each pass later in more detail. *<adbFileName>_iteration_summary.rpt* provides additional details about the saved files.

Iteration Summary Report

The file *<adbFileName>_iteration_summary.rpt* records a summary of how the multiple pass run was invoked either through the GUI or extended_run_shell Tcl script, with arguments for repeating each run. Each new run appears with its own header in the Iteration Summary Report with fields **RUN_NUMBER** and **INVOKED AS** , followed by a

table containing **Seed Index**, corresponding **Seed** value, Comparison data, **Report Analyzed**, and **Saved Design** information.

Example

The first header displays information about the first run invoked from the command line (extended_run_shell). The second run was invoked from the Designer GUI (extended_run_gui).

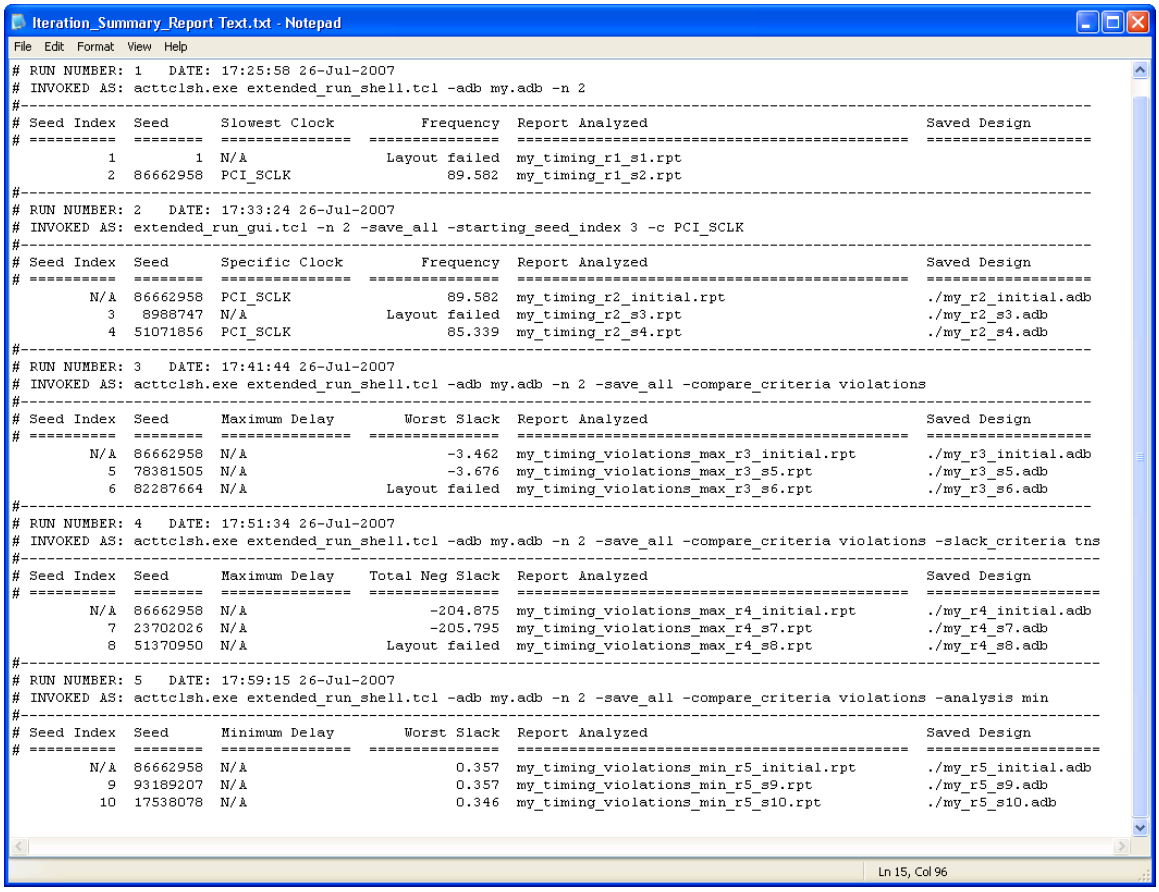


Figure 4 · Iteration Summary Report

See Also

Running Layout

extended_run_shell

extended_run_gui

Analyzing Timing in Your Design

You can perform timing analysis using the SmartTime or Timer tool.

The following table lists the timing tool you can use for your device family.

Family	SmartTime	Timer
Fusion	X	
IGLOO		
IGLOOe	X	
ProASIC3	X	
ProASIC3E	X	
ProASICPLUS	X	
Axcelerator	X	
ProASIC	X	
eX	X	
SX-A	X	
SX		X
MX		X
3200DX		X
ACT3		X
ACT2 /1200XL		X
ACT1		X

For details on SmartTime, refer to the SmartTime online help.

For details on Timer, refer to the Timer online help.

Analyzing Power Consumption in Your Design

Use the SmartPower tool to analyze your designs power consumption. Use the SmartPower tool to:

- Define clock domains
- Specify individual pin frequencies
- View detailed hierarchical analysis of your design
- View global power consumption at the design level

If you wish, you may also view the equations that SmartPower uses to calculate your power consumption.

Viewing Your Netlist

The NetlistViewer tool displays the contents of the design as a schematic, making it easier for you to debug your design. With NetlistViewer, you can view nets, ports, and instances in the schematic view. You can also isolate specific sections of your netlist to simplify your analysis and cross probe with other tools.

There are two versions of the NetlistViewer tool: NetlistViewer in MultiView Navigator (MVN) and NetlistViewer (non-MVN). Which version you use depends on which family you are designing for.

- NetlistViewer in MultiView Navigator supports IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, Axcelerator, eX, and SX-A families.
- NetlistViewer (non-MVN) supports MX, DX, ACT3, ACT2, and ACT1 families.

Used with PinEditor in MultiView Navigator, ChipPlanner, or SmartTime, NetlistViewer in MultiView Navigator assists you in meeting area and timing goals by helping you with critical path identification. NetlistViewer (non-MVN) can also be used alone or with PinEditor Standalone, ChipEditor, or Timer.

When you open your design (.adb) file, Designer will automatically present you with the appropriate tools in the Design Flow window. You must compile your design before you can open it in NetlistViewer.

See Also

Overview- MultiView Navigator

About NetlistViewer in MultiView Navigator

About NetlistViewer (non-MVN)

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.

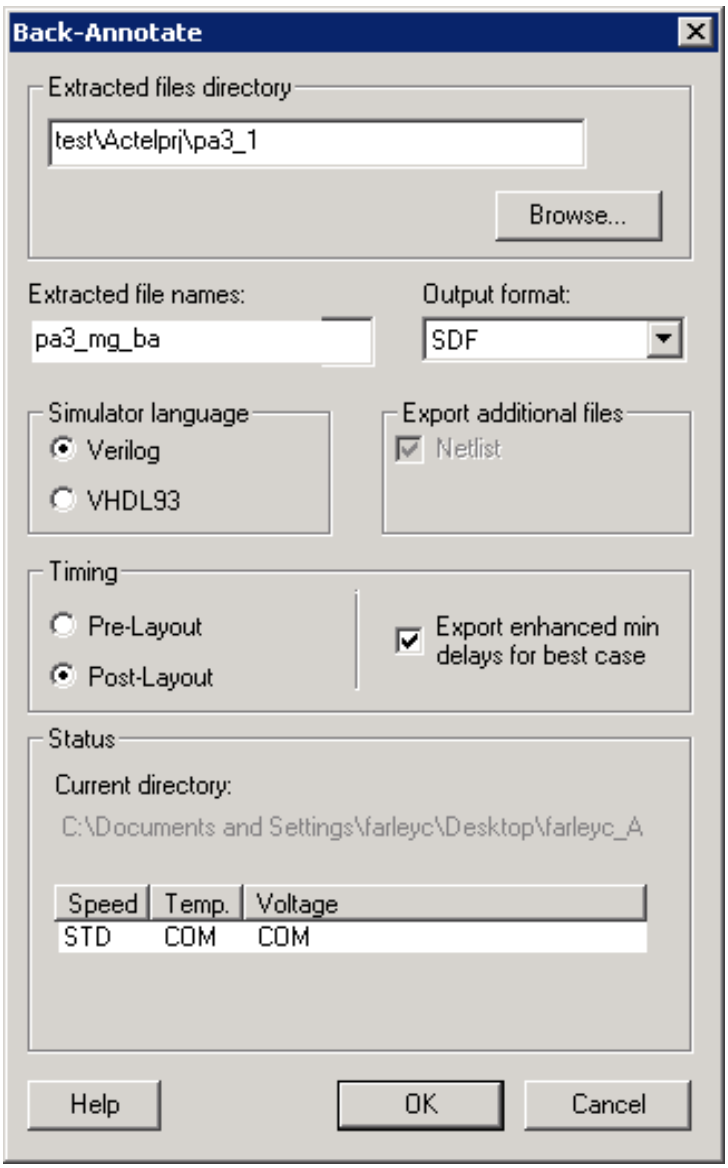


Figure 5 · Back-Annotate Dialog Box

The Back-Annotation command creates the files necessary for back-annotation to the CAE file output type that you choose. 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, choose **Back-Annotate**, or select the Back-Annotate button in the Design Flow window.
2. Make your selections in the Back-Annotate 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. The Extracted File Names field is disabled when you open Designer from the Libero IDE. If you wish to change your extracted file name, you must change the name of your file (**Save As**) and save it outside of the Libero IDE folder structure.

Output Formats: Select the file format of the timing file, SDF or STF.

(STF is only supported for DX, MX, and SX).

Simulator Language: Select either Verilog or VHDL93.

Export Additional Files: Check **Netlist** or **Pin** to export these files at the same time. For Fusion, IGLOO, ProASIC3, and Axcelerator familes, you must export and use the "flattened" netlist (AFL-style) with the back-annotated timing file (SDF) in timing simulation.

Timing - Pre-Layout or Post-Layout sets whether you want to export your pre- or post-layout timing files for back-annotation. You can use pre-layout to backannotate your netlist before place-and-route, but it is less precise than post-layout timing. Post-layout is more precise because it contains your actual design implementation.

Export enhanced min delays for best case - This option changes the best-case number in the SDF file. By default, best case numbers are derived from typical operating conditions. If you enable this option, best case numbers also reflect minimum delay, including variations in process and die.

Note: For IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S families you cannot select SDF format using File > Export Files > Timing.

You must export the netlist from the back-annotate command for IGLOO, Fusion, ProASIC3 and Axcelerator. This selection is hard-coded to be ON. For all other families, the export-netlist and back-annotate actions generate equivalent netlist files, so the back-annotate command does not enforce the writing out of the netlist during back-annotate.

Report Types

You can generate the following types of reports in Designer:

Report Type	Report	Supported Families	Report Contents
Status	Status	All	Provides information about Designer, Device Data, and variable settings for the design.
Timing	Timer (using SmartTime)	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	Provides a text report with the timing information organized by clock domain as in the GUI.
	Timer (using Timer)	SX, MX, 3200DX, ACT3, ACT2, and ACT1	Displays summarized timing delays for paths.
	Bottleneck	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	Creates a report containing information about the bottlenecks in the design.
	Constraints Coverage	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	Creates a report containing information about the constraints in the design.
	Datasheet	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	Creates a report containing information about the external characteristics of the design.
	Timing Violations (using SmartTime)	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	Provides a flat slack report centered around constraint violations.
	Combinational Loops	IGLOO, Fusion, ProASIC3, Axcelerator, RTAX-S, eX, and SX-A	This report displays all loops found during initialization and reports pins associated with the loop(s), and the location where the loop is broken.

Report Type	Report	Supported Families	Report Contents
Resources	Pin	All	Creates 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.
	Flip Flop	All	Creates 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. The flip-flop report can be of two types: Summary or Extended. Both types of 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, modifying, saving, and printing.
IOBank	IO Bank	IGLOO, Fusion, ProASIC3 and Axcelerator	Provides information on the I/O functionality, I/O technologies, I/O banks and I/O voltages.
Power	Power	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator	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.
	Power Cycle Accurate	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, and RTAX-S	Creates a report containing a power waveform with one power value per clock period of half-period instead of an average power for the whole simulation.
	Power Activity and Hazards	IGLOO, Fusion, ProASIC3,	Creates a report containing information about transitions and hazards for each

Report Type	Report	Supported Families	Report Contents
		ProASIC^{PLUS}, ProASIC, Axcelerator, and RTAX-S	clock cycle of the VCD file.
	Power Scenario	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, and RTAX-S	Creates a report containing information about the average power consumption and battery life for a sequence of previously defined operating modes.
Global	CCC Configuration	IGLOO, Fusion, and ProASIC3	The Fusion Dynamic CCC (DYNCCC) and ProASIC3E Dynamic CCC prints out all the values of the configuration pins in a report. You can use these to specify the bitstream that can be shifted in via the shift register.
	GlobalNet	ProASIC^{PLUS}, ProASIC	Creates a report containing information about the net(s) that are assigned or routed using Global or LocalClock resources.
	Global Usage	ProASIC^{PLUS}, and ProASIC	Provides information about the net(s) that are assigned or routed using Global or LocalClock resources.
Block	Block Report	IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S	Creates a report specifically for Blocks and their implementation. Includes Compile, Datasheet, Global, and Interface information.

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:

From the **Tools** menu, choose **Reports** > **Status**. The status report opens in a separate window. You can save or print the report.

Generating a Timing Report

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

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

To generate a timing report:

1. From the Designer **Tools** menu, choose **Reports > Timing > Timer**. The Timing Report Options dialog box appears.
2. Select the options you want to include in the report, and then click **OK**.

The timing report appears in a separate window.

You can also generate the timing report from within SmartTime. From the **Tools** menu, choose **Reports > Report Paths**.

See Also

Understanding timing report

Timing Report Options dialog box

Generating a Timing Violation Report

The timing violations report provides a flat slack report centered around constraint violations.

To generate a timing violation report

1. From the Designer **Tools** menu, choose **Reports > Timing > Timing Violations**. The Timing Violation Report Options dialog box appears.
2. Select the options you want to include in the report, and then click **OK**. The timing violations report appears in a separate window.

You can also generate the timing violations report from within SmartTime. From the **Tools** menu, choose **Reports > Report Violations**.

See Also

Understanding timing violation report

Timing Violations Report Options dialog box

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. From the Tools menu, choose **Reports > Resources > Pin**. This displays the Pin Report dialog box.
2. Select **Number** or **Name** from the List By pull-down menu, then click **OK**. This displays the pin report.

Flip-Flop Reports

The flip-flop report enables you to create a report that lists the number and type of flip-flops used in a design.

In ProASIC3: CC macros in the report are multi-tile flip-flops, which are flip-flops, made of 2 or 4 tiles

All other families: CC macros are flip-flops made of 2 combinatorial macros.

You can generate two types of reports - Summary or Extended:

A Summary report displays whether the flip-flop is a sequential, or multi-tile 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 lists the names of the macros in the design individually.

To generate a flip-flop report:

1. From the **Tools** menu, choose Reports > Resources > FlipFlop. This displays the Flip-Flop Report Options dialog box.
2. Select Summary or Extended from the Type pull-down menu, then click **OK**. This displays the report in a separate window.

I/O Bank Reports

The I/O Bank report provides information about the I/O functionality, I/O technologies, I/O banks and I/O voltages.

To generate the I/O bank report:

From the **Tools** menu, choose **Report > IOBank**. The IOBank report opens in a separate window. You can save or print the report.

The following section shows an excerpt from the I/O Bank report:

I/O Function:

Type	w/o register	w/ register	w/ DDR register
Input I/O	20	0	0
Output I/O	17	0	0
Bidirectional I/O	1	0	0
Differential Input I/O Pairs	0	0	0
Differential Output I/O Pairs	0	0	0

I/O Technology:

	Voltages		I/Os		
I/O Standard(s)	Vcci	Vref	Input	Output	Bidirectional
LVTTL	3.30v	N/A	20	17	1

I/O Bank Resource Usage:

	Voltages		Single I/Os		Diff I/O Pairs		Vref I/Os		
	Vcci	Vref	Used	Total	Used	Total	Used	Total	Vref Pins
Bank0	3.30v	N/A	0	25	0	12	N/A	N/A	N/A
Bank1	3.30v	N/A	0	15	0	7	N/A	N/A	N/A
Bank2	3.30v	N/A	0	17	0	6	N/A	N/A	N/A
Bank3	3.30v	N/A	0	16	0	7	N/A	N/A	N/A
Bank4	3.30v	N/A	0	15	0	7	N/A	N/A	N/A
Bank5	3.30v	N/A	0	22	0	10	N/A	N/A	N/A
Bank6	3.30v	N/A	0	19	0	9	N/A	N/A	N/A
Bank7	3.30v	N/A	0	18	0	7	N/A	N/A	N/A

I/O Voltage Usage:

Voltages		I/Os	
Vcci	Vref	Used	Total
3.30v	N/A	38	147

I/O Functionality

This section of the report indicates the total number of regular input, output, and bidirectional signals. This also shows the differential input and output in the design. The I/O categories are: regular I/Os, registered I/Os or DDR I/Os in the current design.

I/O Technologies

This section of the report specifies the VCCI and VREF voltage requirements for each I/O standard and the number of user I/Os per I/O standards used in the current design.

Note: For voltage referenced I/O standards, input and bidirectional I/Os require both a VCCI and a VREF power supply whereas output I/Os only require a VCCI power supply. This is why these two categories are shown separately in this section.

I/O Banks

This section of the report specifies the following I/O bank characteristics:

- VCCI and VREF voltages assigned for each bank.
- Total number of single-ended I/Os available and used for each bank.
- Total number of differential I/O pairs available and used for each bank.
- Total number of VREF pins assigned for each bank. This information is only relevant if a bank has been assigned a VREF voltage.
- Total number of voltage referenced I/Os available and used for each bank. Voltage referenced I/Os are available only if VREF pins have been assigned.

I/O Voltages

This section of the report indicates the current design voltage requirements.

For each VCCI and VCCI/VREF user I/O demand in the current design, this table reports the total number of bonded I/Os available on the device that satisfy this demand.

Note: For an I/O bank assignment (VCCI and VREF assignment) to be valid for the current design, the I/O voltage table must show no violation. Violations are indicated with an asterisk ("*") when the number of user I/Os that need a given VCCI or VCCI/VREF assignment is less than the total number of bonded I/Os that can satisfy this demand.

Power Reports

The power report enables you to quickly determine if any power consumption problems exist in your design.

To generate a power report:

1. From the Designer **Tools** menu, choose **Reports > Power > Power**. The Power Report dialog box appears.
2. Select the options you want to include in the report, and then click **OK**. The power report appears in a separate window.

You can also generate the power report from within SmartPower. From the **Tools** menu, choose **Reports > Power Report**, or click the **Report** button to open the **Report** dialog box. By default, the report includes global design information and a power summary. Specify which results you want to display by selecting the categories and their options.

The power report dialog box is organized in the following panels: General, Operating Conditions, Options, Breakdown by Instance, Activity Summary, and Enable Rates Summary.

General

The general panel enables you to select what to include in the report, the report format, and the mode that you want to generate the report for.

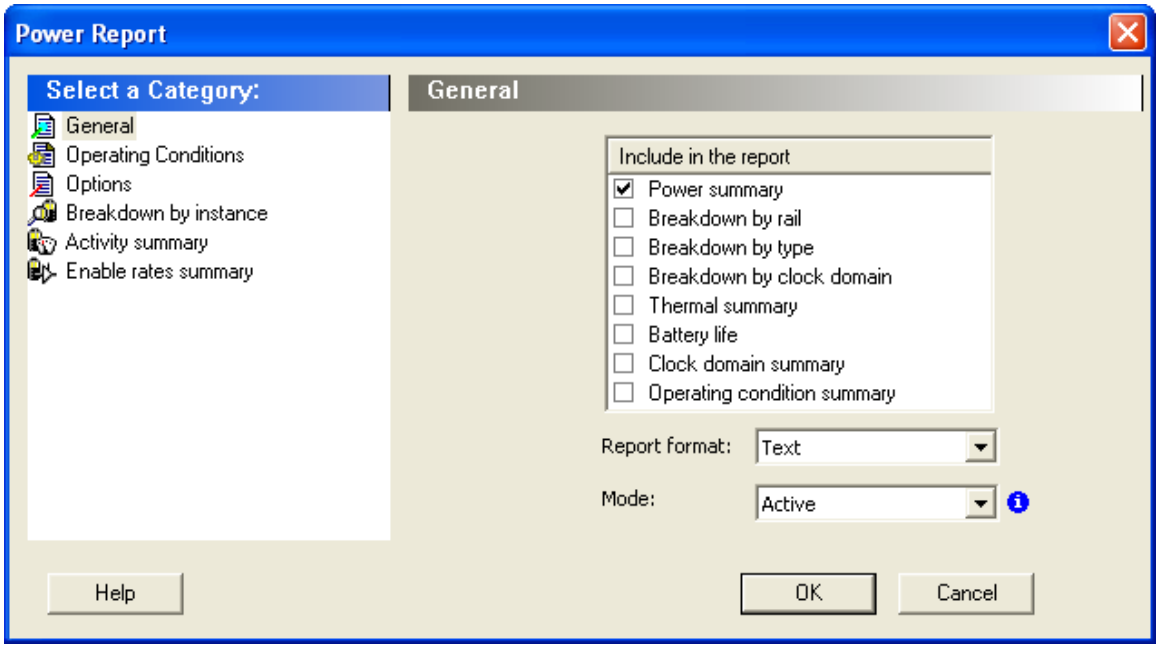


Figure 6 · SmartPower Power Report Dialog Box - General Panel

Include in the Report

Select the option(s) that you want to include in the Power Report:

Power summary – This section reports the static, dynamic and total power consumption of the design.

Breakdown by rail – This section shows the power consumption of each rail.

Breakdown by type – This section enables reporting on the power consumption according to: gates, nets, clocks, core static, IOs and memories.

Breakdown by clock domain – This section enables reporting on the power consumption of each clock domain.

Thermal summary – This section includes a thermal report. The ambient temperature can be defined by the operating conditions or defined by the ambient temperature.

When the first option is selected, the following characteristics are reported:

- Operating conditions
- Temperature range
- Junction temperature

When the second option is selected, the following characteristics are reported:

- Ambient temperature
- Cooling style
- Package
- Thermal resistance Theta-JA

- Junction temperature
- Temperature range
- Junction temperature range limits specification

Battery Life – This section reports the battery life.

Clock Domain Summary – This section reports the clock and data frequencies for each clock domain.

Operating Condition Summary – This section reports the operating conditions.

Report Format

Select **Text** or **CSV** (Comma Separated Value) as the desired export format.

Mode

Select a mode to generate the report for.

Operating Conditions

The Operating Conditions panel enables you to select the operating conditions for the current report.

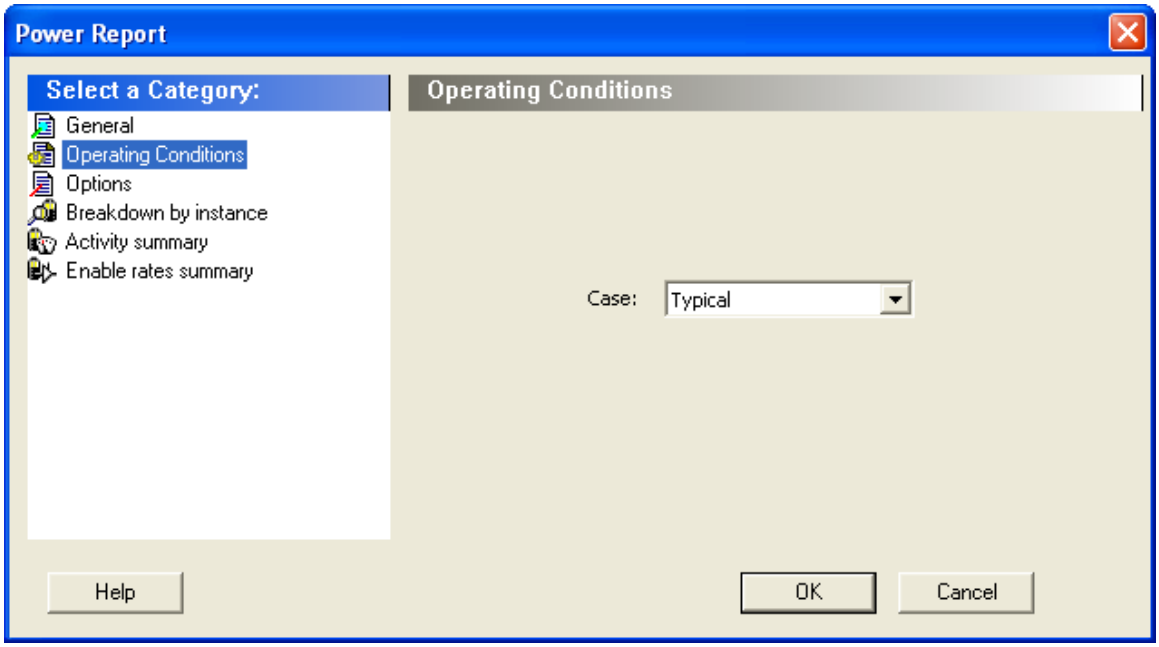


Figure 7 · SmartPower Power Report Dialog Box - Operating Conditions Panel

Options

The Options panel enables you to select power and frequency units and to use toggle rates.

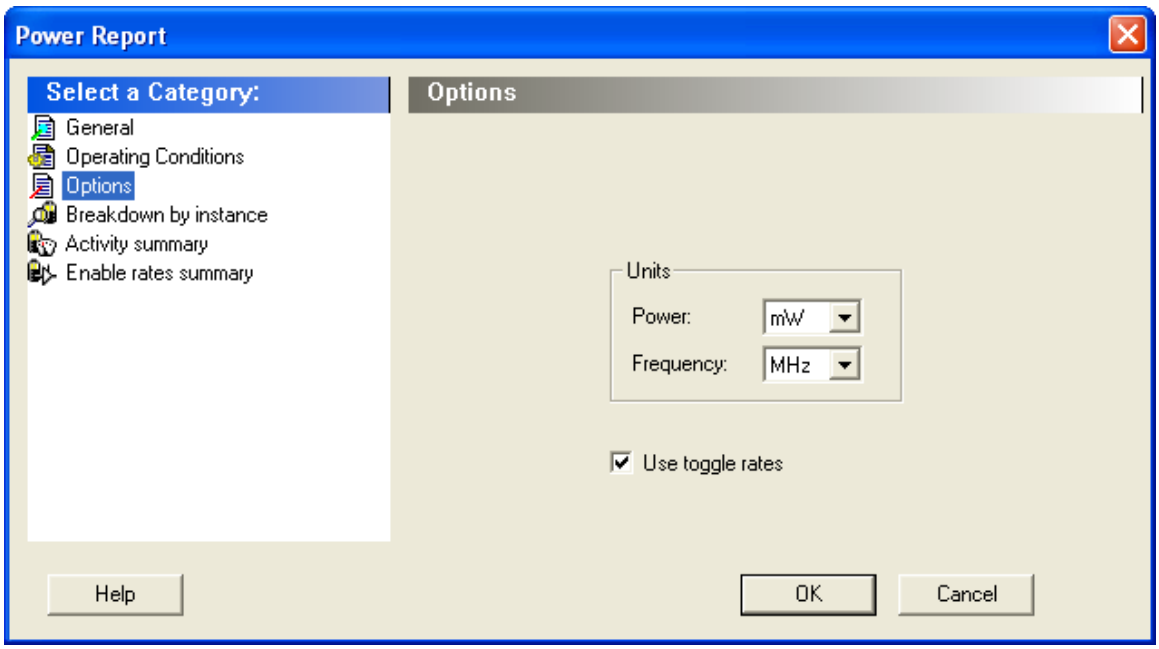


Figure 8 · SmartPower Power Report Dialog Box - Options Panel

Units

Frequency: Sets unit preferences for frequency - Hz, KHz, MHz.

Power Units: Sets unit preferences for power - W, mW, or uW.

Use Toggle Rates

When toggle rates are active (**Use Toggle Rates** box is checked), the data frequency of all the clock domains is defined as a function of the clock frequency. This updates the data frequency automatically when you update the clock frequency. Toggle rates enable you to specify the data frequency as a percentage of clock frequency, but you can no longer specify the data frequency as a number, only as a percentage of the clock frequency. To set the data frequency again, clear the **Use Toggle Rates** option.

Breakdown by Instance

From this panel you can include the breakdown by instance in the report and set specific options.

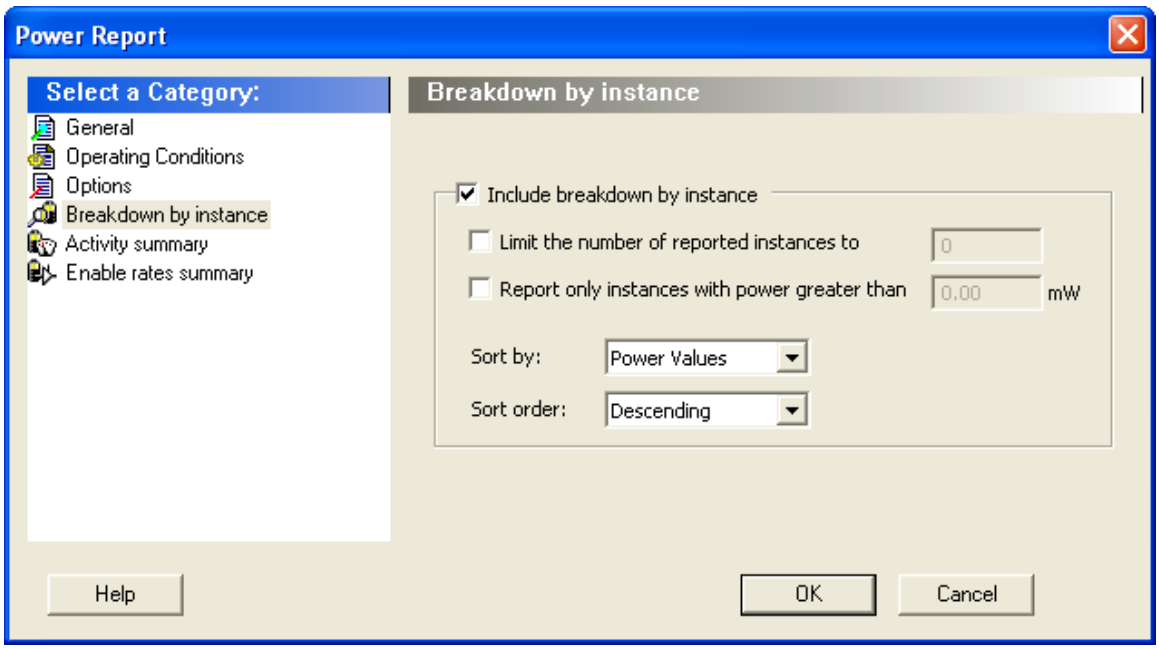


Figure 9 · SmartPower Power Report Dialog Box - Breakdown By Instance Panel

Include breakdown by instance – This section shows the power consumption of each element that has been instantiated in the design: gates, nets, memories, and IOs.

The breakdown by instance can be filtered by:

- **Limit the number of reported instances to:** will limit the number of instances reported to the specified number.
- **Report only instances with power greater than:** any instance with power consumption below the selected value will not be reported.

This section can be sorted by selecting the preferred method:

- **Sort by:** Name (alphabetical) or Power Values
- **Sort Order:** Ascending or Descending

Note: The filter reduces the number of lines in the report, one per instance.

Activity Summary

From this panel you can include the activity summary in the report and set specific options.

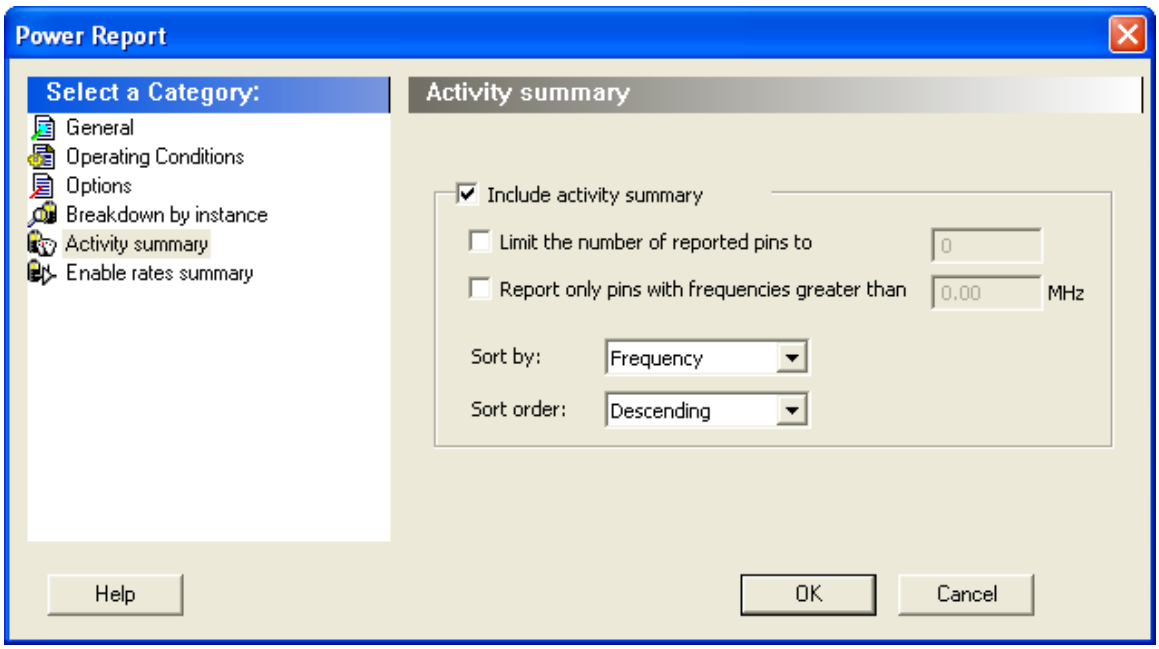


Figure 10 · SmartPower Power Report Dialog Box - Activity Summary Panel

Include activity summary – This section shows the activity summary and reports the pin, net, domain, frequency, and frequency source for each pin.

The activity summary can be filtered by:

- **Limit the number of reported pins to:** will limit the number of pins reported to the specified number.
- **Report only pins with frequencies greater than:** any pin with a frequency below the selected value will not be reported.

This section can be sorted by selecting the preferred method:

- **Sort by:** Pin Name, Domain, Frequency, or Source
- **Sort Order:** Ascending or Descending

Enable Rates Summary

From this panel you can include the enable rates summary in the report and set specific options.

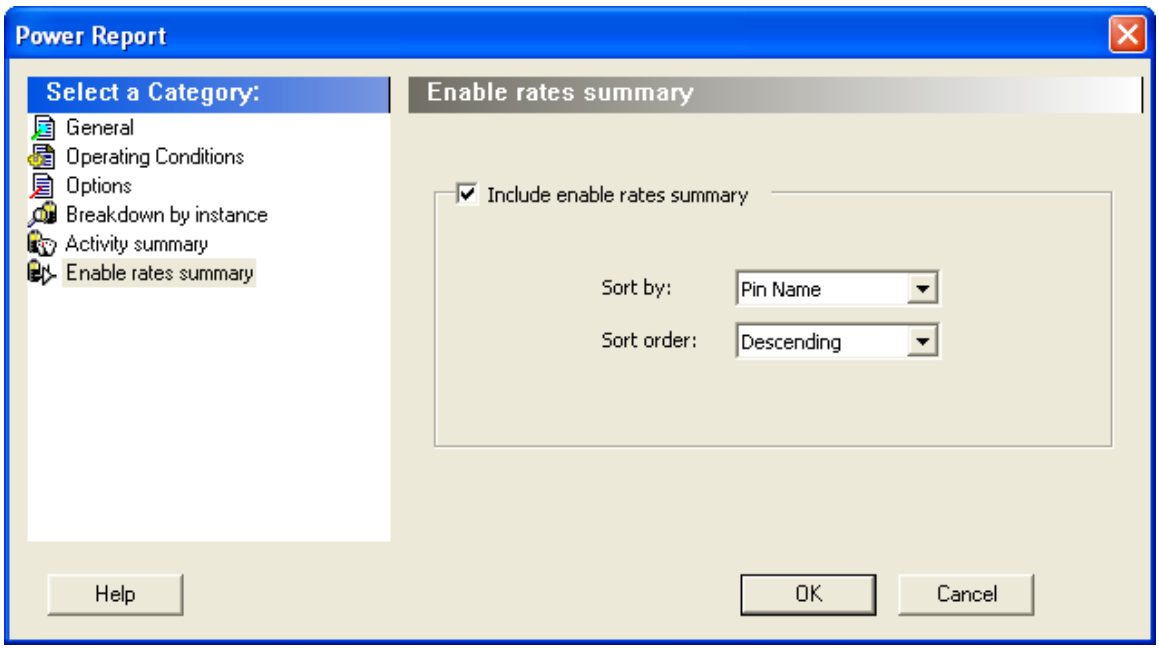


Figure 11 · SmartPower Power Report Dialog Box - Enable Rates Summary Panel

Include enable rates summary – This section shows the enable rates summary and reports the driver, net, rate, source. and type for each pin.

This section can be sorted by selecting the preferred method:

- **Sort by:** Driver, Net, Rate, Source, or Type
- **Sort Order:** Ascending or Descending

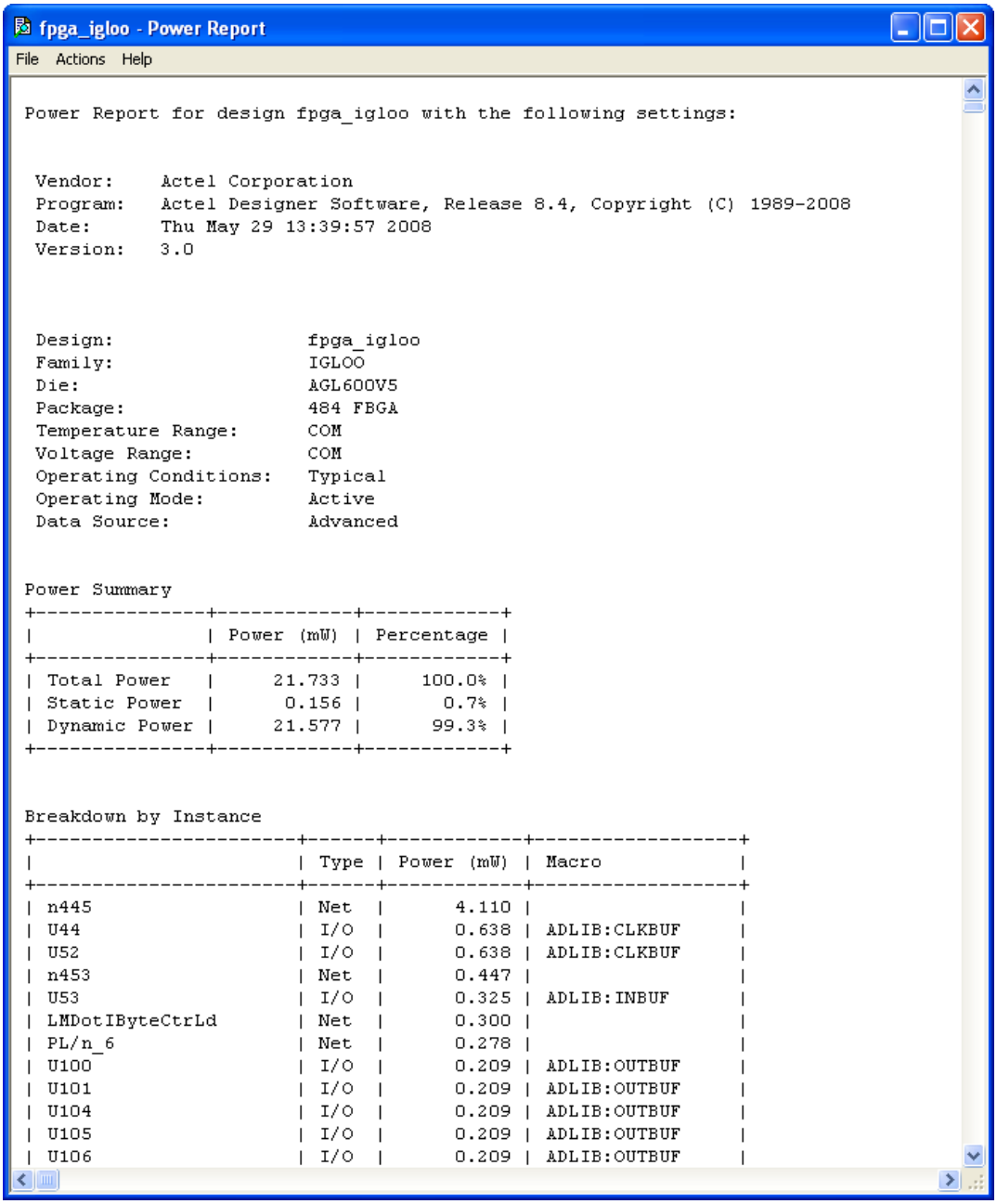


Figure 12 · SmartPower Report

In addition to the information selected on the **Power Reports** dialog box, the report contains global design information.

Global design information: This section shows the target family, the package and the die. It also shows information about the operating conditions, speed grade and power mode. This option is set by default.

See Also

Report (Power)

Cycle-Accurate Power Reports

Traditional power analysis based on a VCD simulation file reports an average power for the entire simulation time. Based on a VCD simulation file, the cycle-accurate power analysis will report one power value per clock period (or half-period) instead of an average power for the whole simulation. This feature allows to easily identify the worst cycle in terms of power performance; and helps to understand and further minimize power consumption by facilitating the analysis of data-dependent power variations, as well as dynamic power variations due to clock-gating, or even clock frequency variations.

To generate a cycle-accurate power report:

1. From the Designer **Tools** menu, choose **Reports > Power > Power Cycle Accurate**. The Cycle Accurate Power Report dialog box appears.
2. Select the options you want to include in the report, and then click **OK**. The cycle accurate power report appears in a separate window.

You can also generate the cycle accurate power report from within SmartPower. From the **Tools** menu, choose **Reports > Cycle Accurate Power Report** to open the cycle accurate power report dialog box; or select a vcd file from the Modes and Scenarios toolbar, and from the right-click menu, select **Tools > Power Cycle Accurate**.

The cycle-accurate power report dialog box is organized in the following panels: General, Sampling Period, Partial Parsing, Top-Level Name, Glitch Filtering, and History Size Reduction.

General

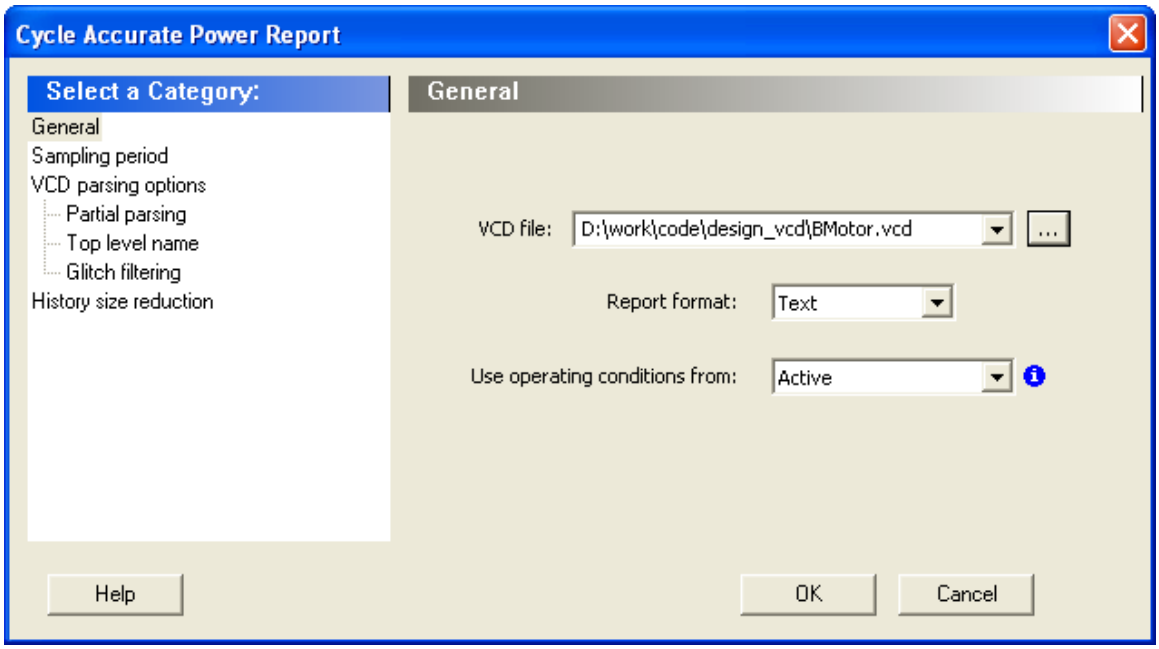


Figure 13 · SmartPower Cycle Accurate Power Report Dialog Box – General

VCD file - Select the VCD file you want to import.

Report format - Select **Text** or **CSV** (Comma Separated Value) as the desired export format.

Use operating conditions from – Select the mode from which the operating conditions will be used.

Sampling Period

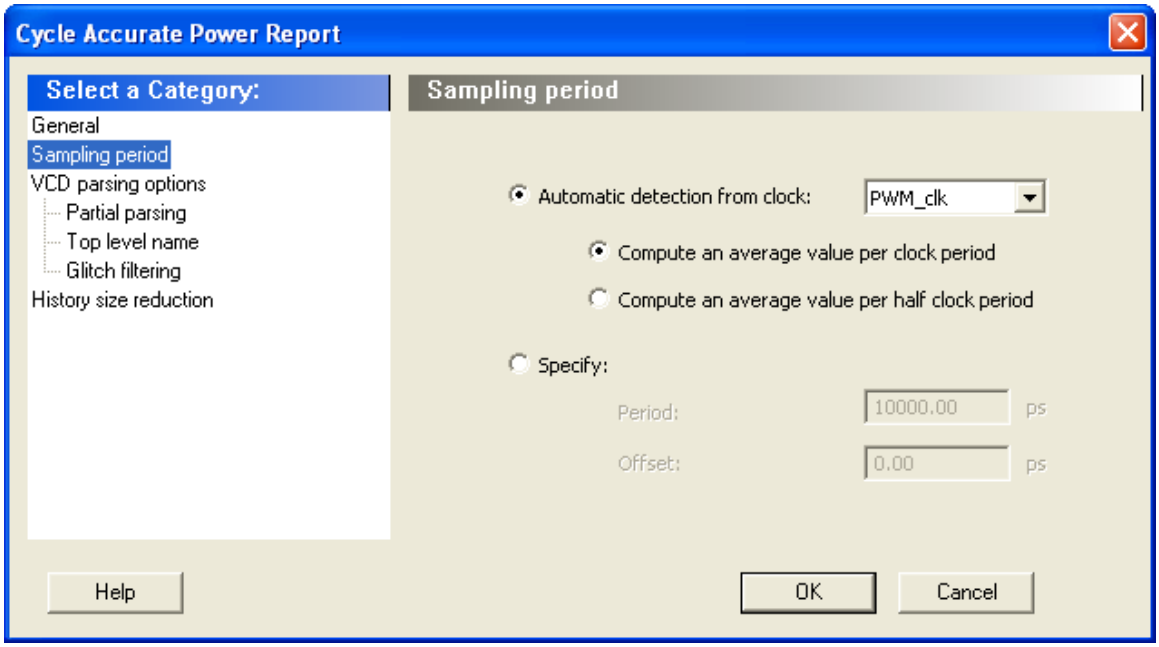


Figure 14 · SmartPower Cycle Accurate Power Report Dialog Box – Sampling Period

Automatic detection from clock – This option automatically detects the sampling period from the fastest clock. You can also select any other clock in your design. You can specify whether the average value is computed per period or half-period.

Specify – Select this option to specify the period and offset used to calculate the sampling period.

Partial Parsing

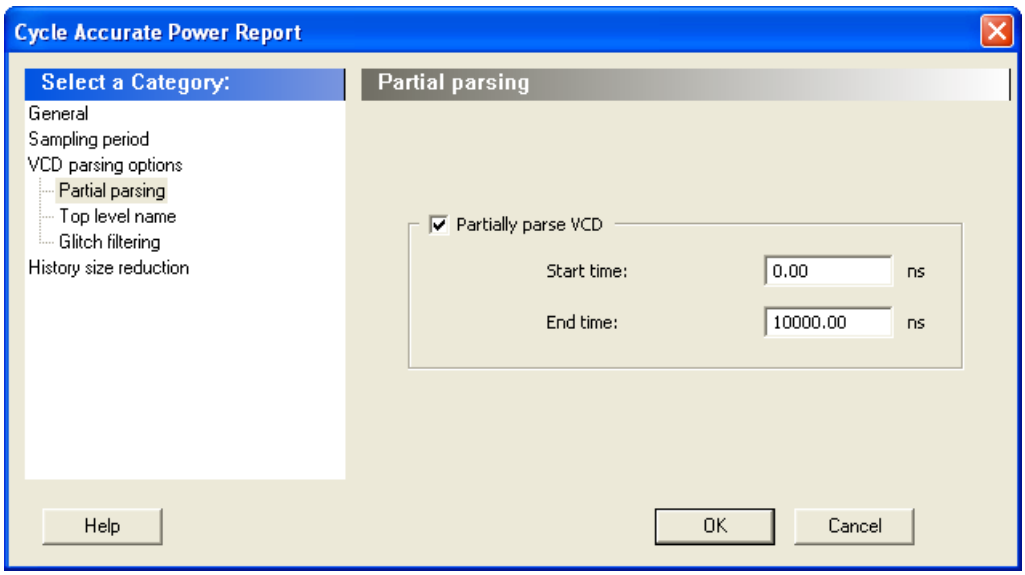


Figure 15 · SmartPower Cycle Accurate Power Report Dialog Box – Partial Parsing

Partially parse VCD file - Specify the Start and End times to partially parse the VCD file. This option can be used for large VCD files.

Top-Level Name

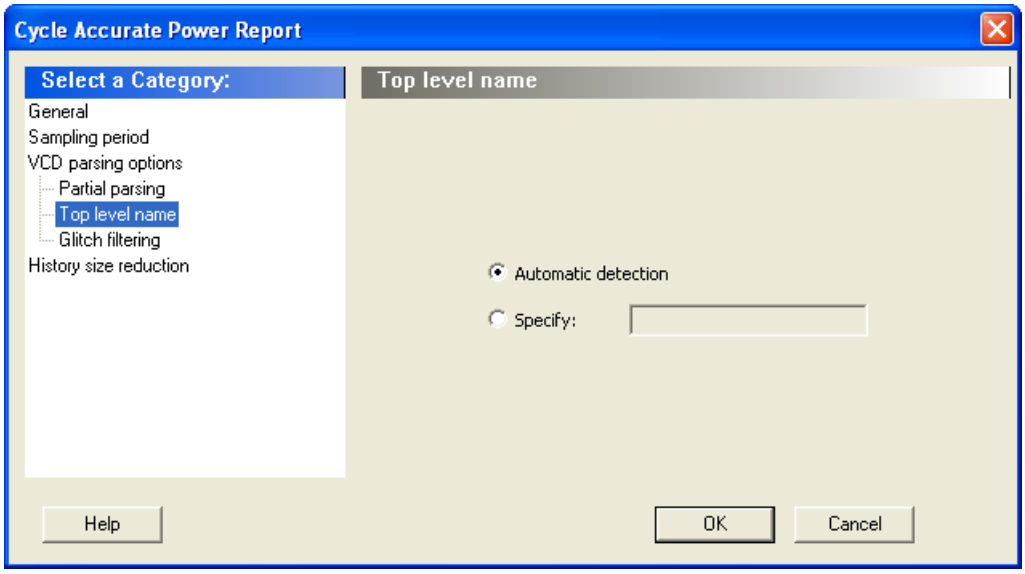


Figure 16 · SmartPower Cycle Accurate Power Report Dialog Box – Top-Level Name

This option enables you to select how the top-level name is specified. Select **Automatic Detection** to let the VCD reader automatically detect the top-level name of the design, or select **Specify** to manually specify the top-level name.

Glitch Filtering

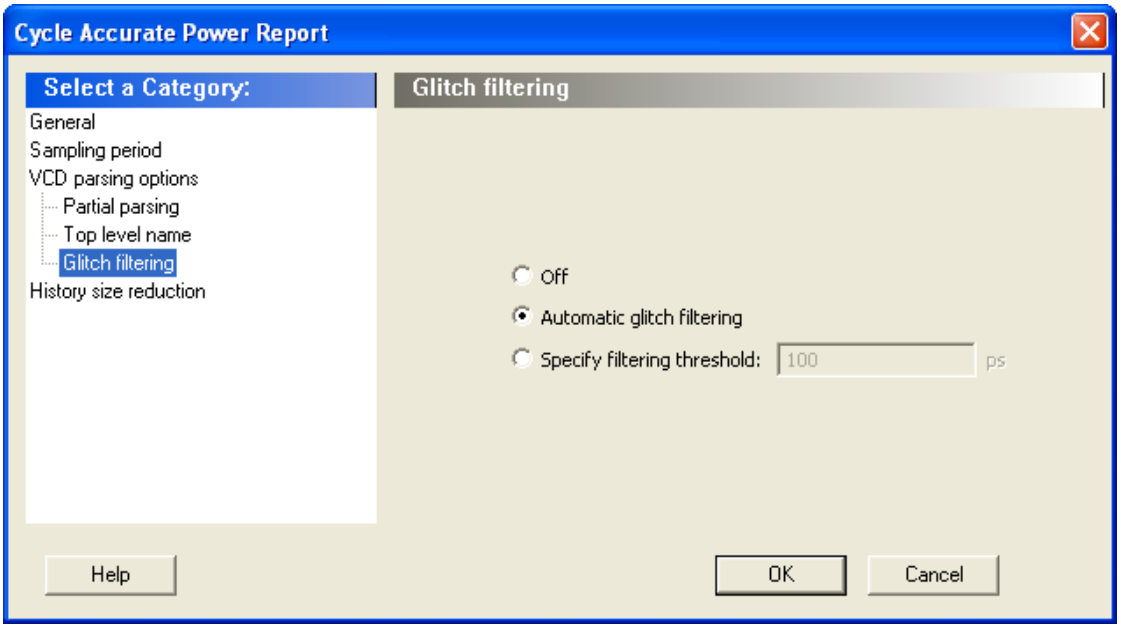


Figure 17 · SmartPower Cycle Accurate Power Report Advanced Options Dialog Box – Glitch Filtering

This panel enables you to filter out pulses of short durations by selecting **Automatic Glitch Filtering** or by entering a value in the **Specify Filtering Threshold** field. The default glitch filtering option is **Automatic Glitch Filtering**.

History Size Reduction

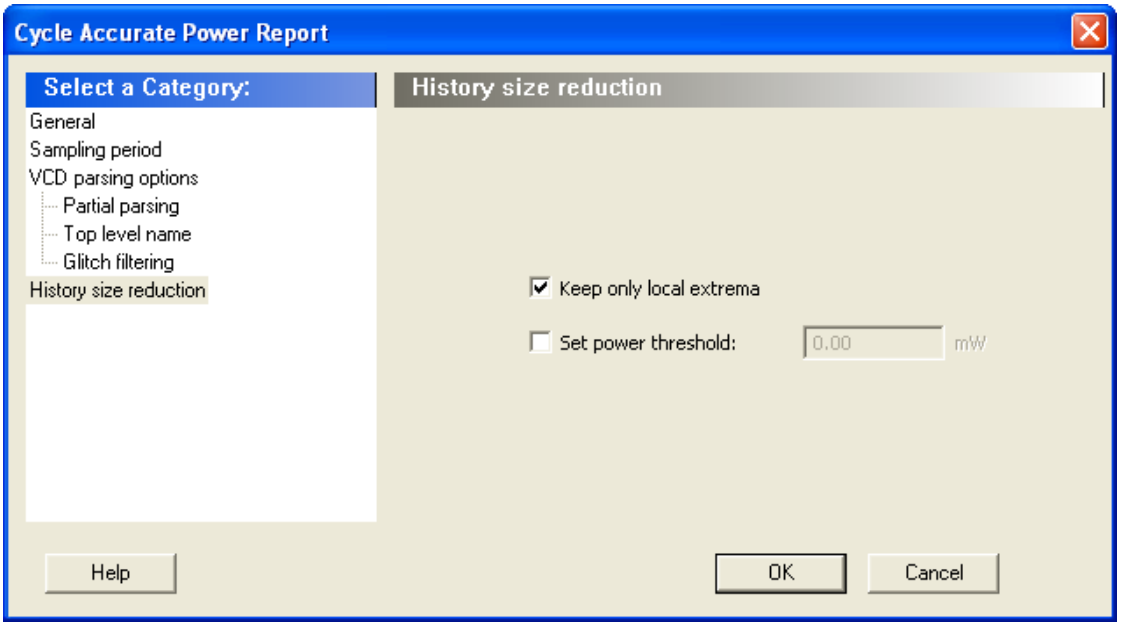
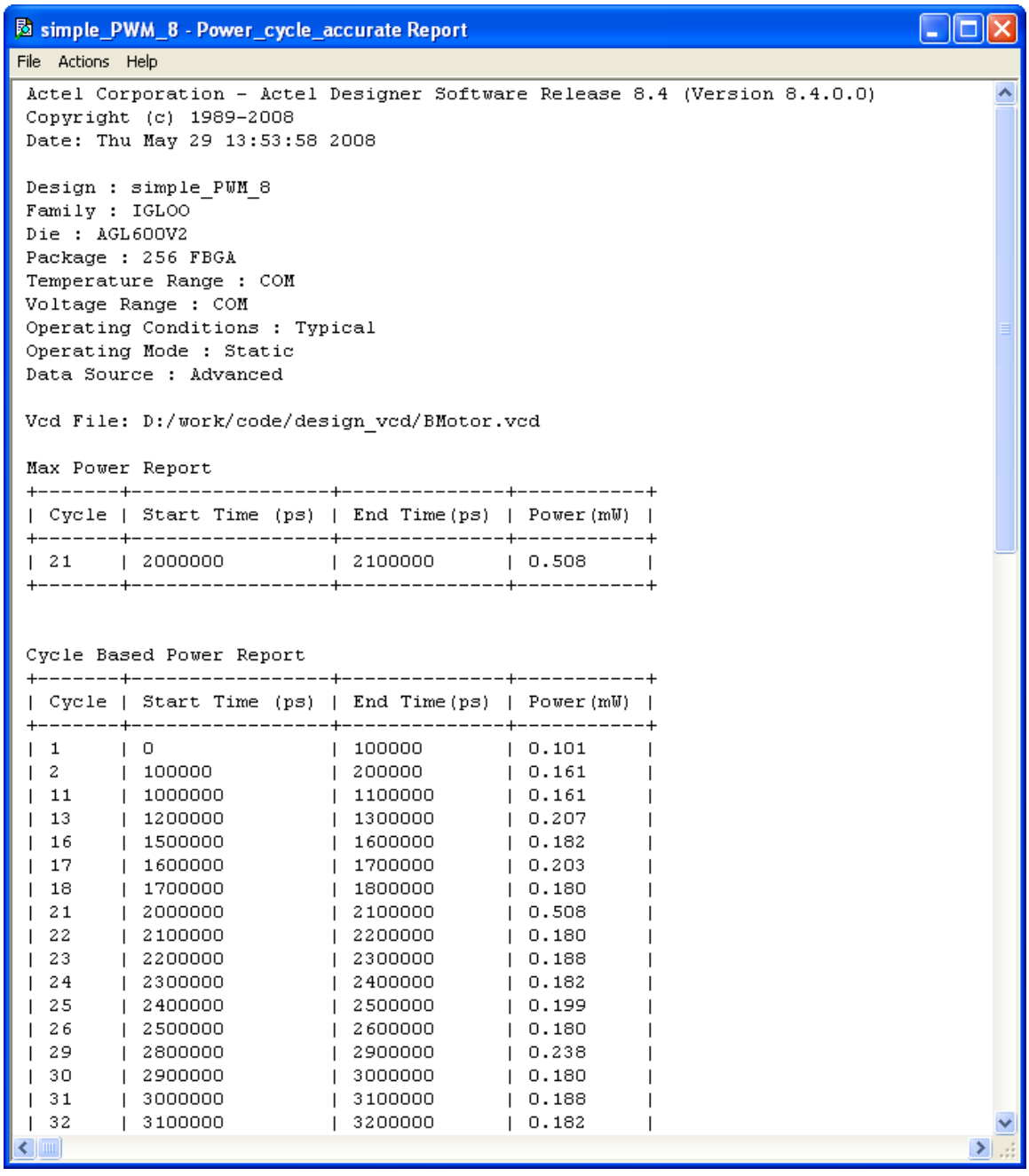


Figure 18 · SmartPower Cycle Accurate Power Report Advanced Options Dialog Box – History Size Reduction

This panel enables you to limit the history size by keeping only local extrema or setting a power threshold.

The results are displayed in the cycle accurate power report below.



simple_PWM_8 - Power_cycle_accurate Report

File Actions Help

```
Actel Corporation - Actel Designer Software Release 8.4 (Version 8.4.0.0)
Copyright (c) 1989-2008
Date: Thu May 29 13:53:58 2008

Design : simple_PWM_8
Family : IGLOO
Die : AGL600V2
Package : 256 FBGA
Temperature Range : COM
Voltage Range : COM
Operating Conditions : Typical
Operating Mode : Static
Data Source : Advanced

Vcd File: D:/work/code/design_vcd/BMotor.vcd

Max Power Report
+-------+-----------------+--------------+-----------+
| Cycle | Start Time (ps) | End Time(ps) | Power(mW) |
+-------+-----------------+--------------+-----------+
| 21    | 2000000         | 2100000      | 0.508     |
+-------+-----------------+--------------+-----------+


Cycle Based Power Report
+-------+-----------------+--------------+-----------+
| Cycle | Start Time (ps) | End Time(ps) | Power(mW) |
+-------+-----------------+--------------+-----------+
| 1     | 0               | 100000       | 0.101     |
| 2     | 100000          | 200000       | 0.161     |
| 11    | 1000000         | 1100000      | 0.161     |
| 13    | 1200000         | 1300000      | 0.207     |
| 16    | 1500000         | 1600000      | 0.182     |
| 17    | 1600000         | 1700000      | 0.203     |
| 18    | 1700000         | 1800000      | 0.180     |
| 21    | 2000000         | 2100000      | 0.508     |
| 22    | 2100000         | 2200000      | 0.180     |
| 23    | 2200000         | 2300000      | 0.188     |
| 24    | 2300000         | 2400000      | 0.182     |
| 25    | 2400000         | 2500000      | 0.199     |
| 26    | 2500000         | 2600000      | 0.180     |
| 29    | 2800000         | 2900000      | 0.238     |
| 30    | 2900000         | 3000000      | 0.180     |
| 31    | 3000000         | 3100000      | 0.188     |
| 32    | 3100000         | 3200000      | 0.182     |
```

Figure 19 · SmartPower Cycle Accurate Power Report

Activity and Hazards Reports

Traditional Power Analysis based on a VCD simulation file reports an average power value that will account for all nets switching in the design. This switching includes functional transitions and spurious transitions. Due to the delay of

each gate, paths arriving at one internal gate may have different propagation delays. Therefore, a gate may exhibit multiple spurious transitions before settling to the correct logic level. The Activity and Hazards Power Report allows to quickly identify gates and nets of the design that consume power because of spurious transitions. This is helpful to understand and further minimize power consumption. The activity and hazards report reads a VCD file and reports transitions and hazards for each clock cycle of the VCD file.

To generate an activity and hazards power report:

1. From the Designer **Tools** menu, choose **Reports > Power > Power Activity and Hazards**. The Activity and Hazards Power Report dialog box appears.
2. Select the options you want to include in the report, and then click **OK**. The activity and hazards report appears in a separate window.

You can also generate the activity and hazards report from within SmartPower. From the **Tools** menu, choose **Reports > Activity and Hazards Report**; or select a vcd file from the Modes and Scenarios toolbar, and from the right-click menu, select **Tools > Power Activity and Hazards**

The activity and hazards report dialog box is organized in the following panels: General, Partial Parsing, Top-Level Name, Glitch Filtering, and Clock Domains.

General

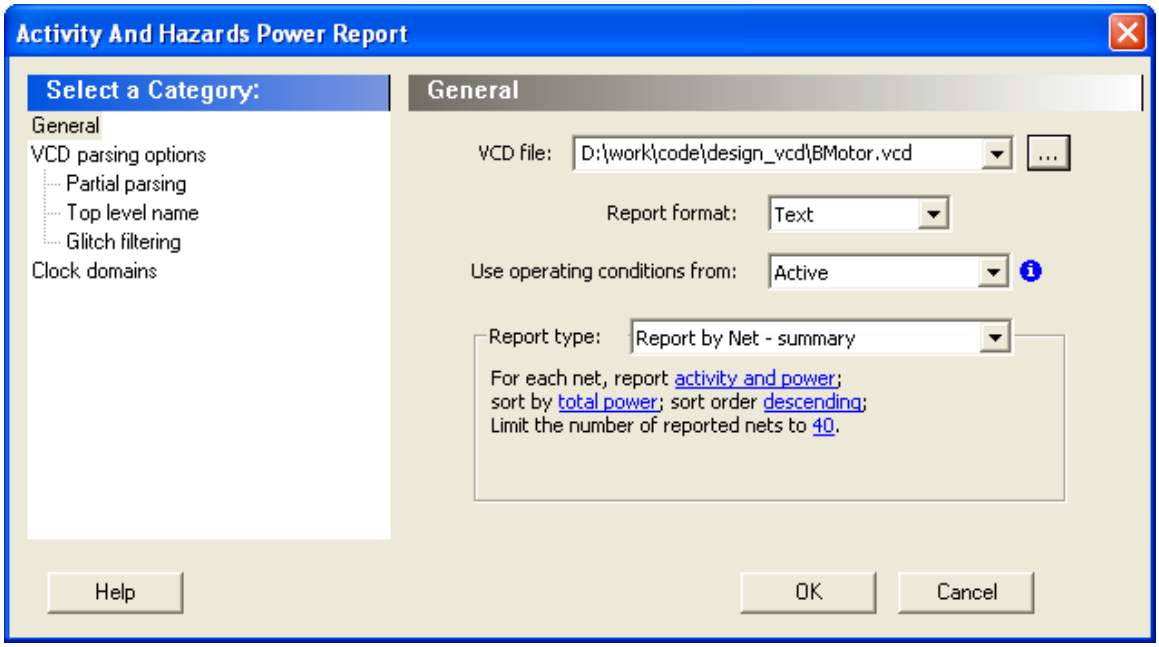


Figure 20 · SmartPower Activity and Hazards Report Dialog Box – General

VCD file - Select the VCD file you want to import.

Report format - Select **Text** or **CSV** (Comma Separated Value) as the desired export format.

Use operating conditions from – Select the mode from which the operating conditions will be used.

Report type - Select the report type:

- Report by net - summary: summary report by net
- Report by net - detailed: detailed report by cycle
- Report by cycle - summary: summary report by net
- Report by cycle - detailed: detailed report by cycle

The selected report type reports activity and power for each net sorted by power in descending order and limits the number of reported nets to 20 by default. To change these options, click each option and from the pop-up menu, select the desired option:

- Report: Select the query report type: activity, power, or activity and power.
- Sort by: Select the query sort by functional power, functional transitions, spurious power, spurious transitions, or total power.
- Sort order: Select the query sort order: ascending or descending.
- Limit the number of reported nets: Enter the query filter limit.

Partial Parsing

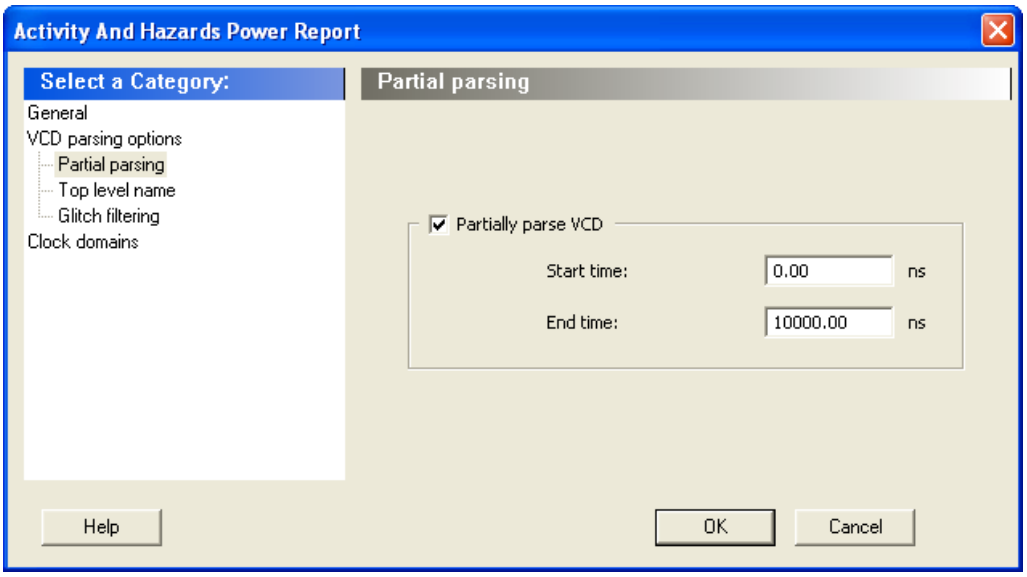


Figure 21 · SmartPower Activity and Hazards Report Dialog Box – Partial Parsing

Partially parse VCD file - Specify the Start and End times to partially parse the VCD file. This option can be used for large VCD files.

Top-Level Name

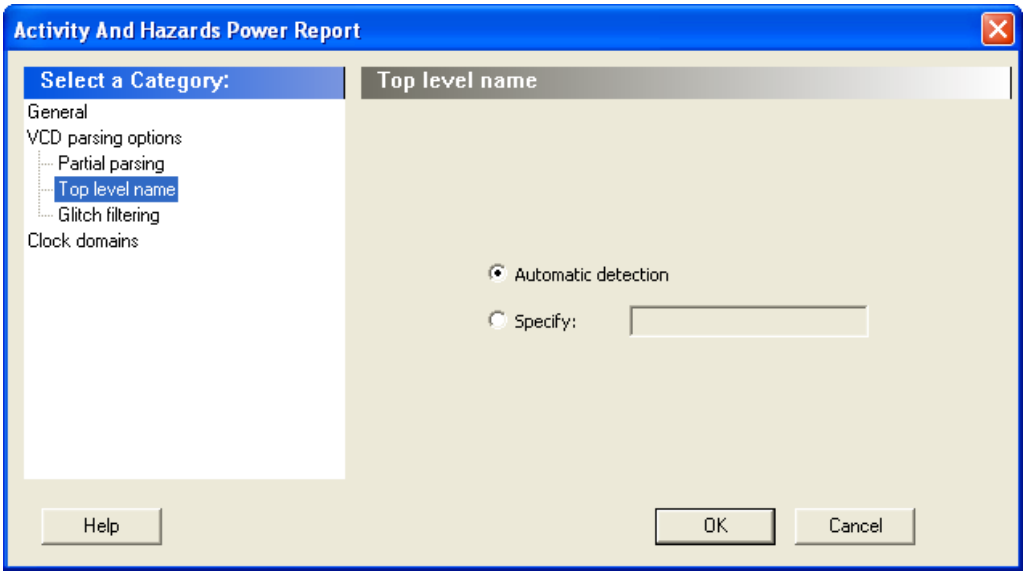


Figure 22 · SmartPower Activity and Hazards Report Dialog Box – Top-Level Name

This option enables you to select how the top-level name is specified. Select **Automatic Detection** to let the VCD reader automatically detect the top-level name of the design, or select **Specify** to manually specify the top-level name.

Glitch Filtering

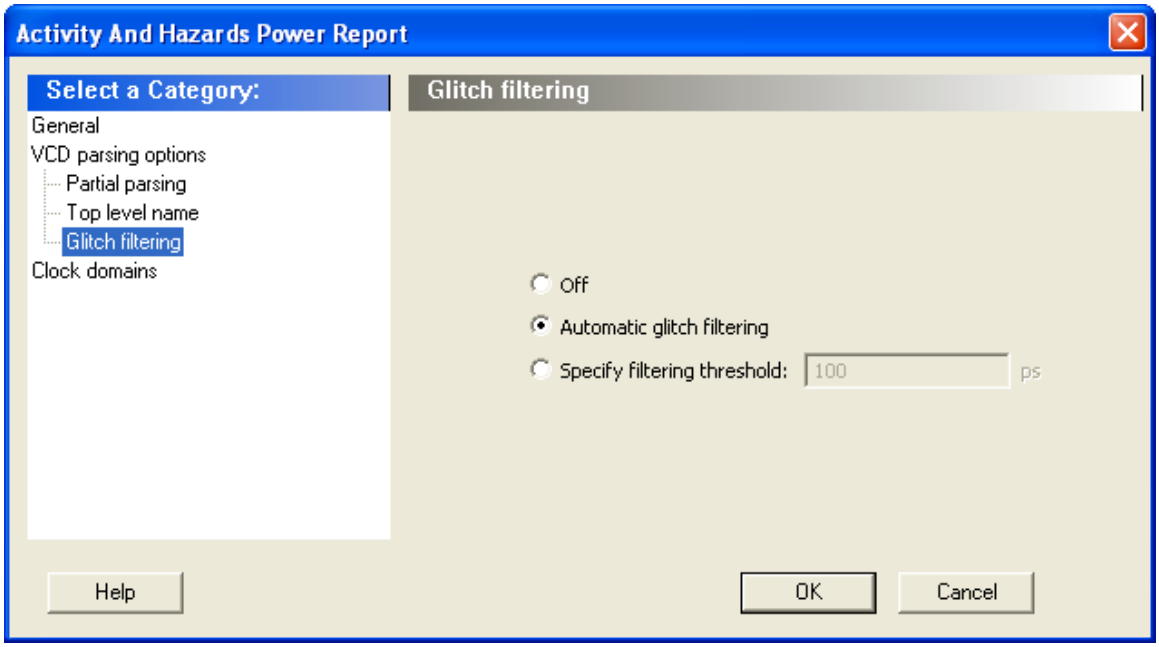


Figure 23 · SmartPower Activity and Hazards Report Advanced Options Dialog Box – Glitch Filtering

This option enables you to filter out pulses of short durations by selecting **Automatic Glitch Filtering** or by entering a value in the **Specify Filtering Threshold** field. The default glitch filtering option is **Automatic Glitch Filtering**.

Clock Domains

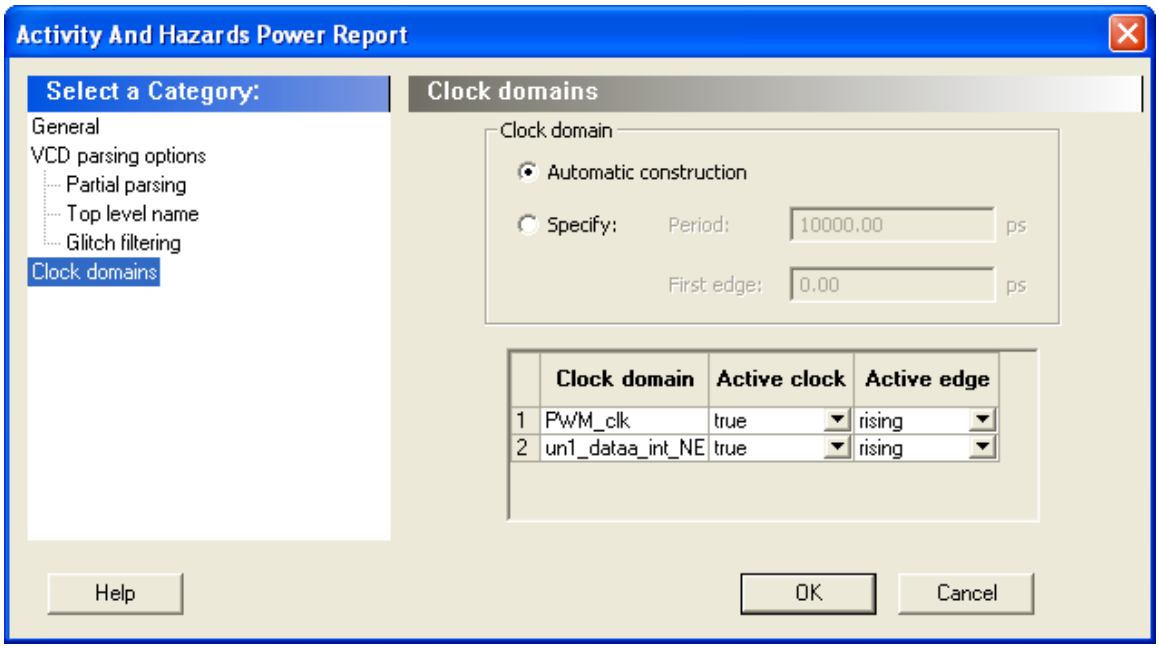


Figure 24 · SmartPower Activity and Hazards Report Advanced Options Dialog Box – Clock Domains

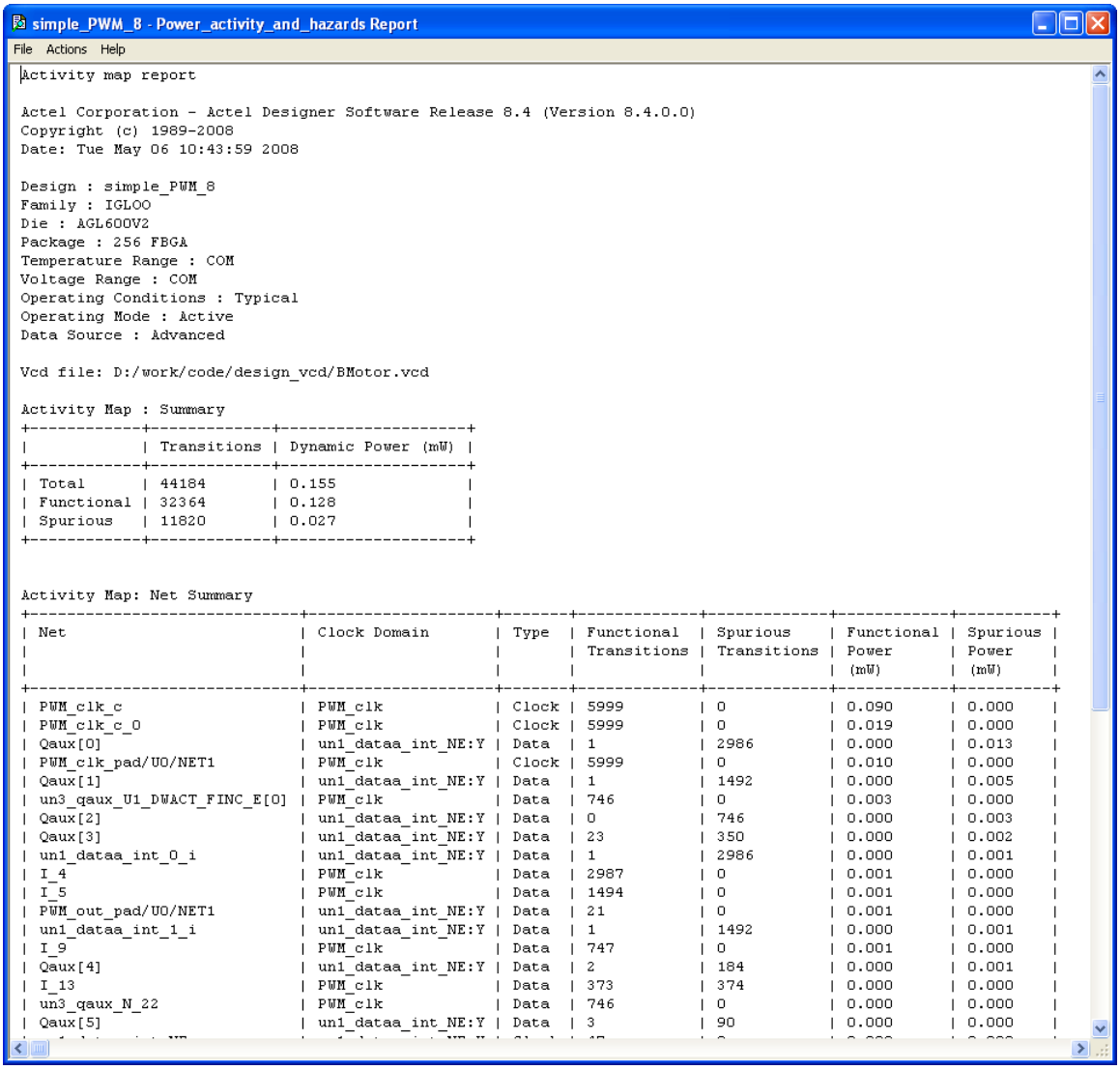
The clock domain can be automatically constructed or it can be specified by the user.

Automatic construction – This option automatically constructs the clock domain. SmartPower automatically analyzes your design to assess if a clock is active and what is the active edge.

Specify – Select this option to specify the period and first edge.

Use the clock domain table to set the active edge (rising, falling or both) and to set a clock as transparent.

The results are displayed in the activity and hazards power report below.



simple_PWM_8 - Power_activity_and_hazards Report

File Actions Help

```
Activity map report

Actel Corporation - Actel Designer Software Release 8.4 (Version 8.4.0.0)
Copyright (c) 1989-2008
Date: Tue May 06 10:43:59 2008

Design : simple_PWM_8
Family : IGLOO
Die : AGL600V2
Package : 256 FBGA
Temperature Range : COM
Voltage Range : COM
Operating Conditions : Typical
Operating Mode : Active
Data Source : Advanced

Vcd file: D:/work/code/design_vcd/BMotor.vcd

Activity Map : Summary
+------------+-------------+--------------------+
|            | Transitions | Dynamic Power (mW) |
+------------+-------------+--------------------+
| Total      | 44184       | 0.155              |
| Functional | 32364       | 0.128              |
| Spurious   | 11820       | 0.027              |
+------------+-------------+--------------------+


Activity Map: Net Summary
+-----------------------------+--------------------+-------+-------------+------------+------------+----------+
| Net                         | Clock Domain       | Type  | Functional  | Spurious   | Functional | Spurious |
|                             |                    |       | Transitions | Transitions| Power      | Power    |
|                             |                    |       |             |            | (mW)       | (mW)     |
+-----------------------------+--------------------+-------+-------------+------------+------------+----------+
| PWM_clk_c                   | PWM_clk            | Clock | 5999        | 0          | 0.090      | 0.000    |
| PWM_clk_c_0                 | PWM_clk            | Clock | 5999        | 0          | 0.019      | 0.000    |
| Qaux[0]                     | un1_dataa_int_NE:Y | Data  | 1           | 2986       | 0.000      | 0.013    |
| PWM_clk_pad/U0/NET1         | PWM_clk            | Clock | 5999        | 0          | 0.010      | 0.000    |
| Qaux[1]                     | un1_dataa_int_NE:Y | Data  | 1           | 1492       | 0.000      | 0.005    |
| un3_qaux_U1_DWACT_FINC_E[0] | PWM_clk            | Data  | 746         | 0          | 0.003      | 0.000    |
| Qaux[2]                     | un1_dataa_int_NE:Y | Data  | 0           | 746        | 0.000      | 0.003    |
| Qaux[3]                     | un1_dataa_int_NE:Y | Data  | 23          | 350        | 0.000      | 0.002    |
| un1_dataa_int_0_i           | un1_dataa_int_NE:Y | Data  | 1           | 2986       | 0.000      | 0.001    |
| I_4                         | PWM_clk            | Data  | 2987        | 0          | 0.001      | 0.000    |
| I_5                         | PWM_clk            | Data  | 1494        | 0          | 0.001      | 0.000    |
| PWM_out_pad/U0/NET1         | un1_dataa_int_NE:Y | Data  | 21          | 0          | 0.001      | 0.000    |
| un1_dataa_int_1_i           | un1_dataa_int_NE:Y | Data  | 1           | 1492       | 0.000      | 0.001    |
| I_9                         | PWM_clk            | Data  | 747         | 0          | 0.001      | 0.000    |
| Qaux[4]                     | un1_dataa_int_NE:Y | Data  | 2           | 184        | 0.000      | 0.001    |
| I_13                        | PWM_clk            | Data  | 373         | 374        | 0.000      | 0.000    |
| un3_qaux_N_22               | PWM_clk            | Data  | 746         | 0          | 0.000      | 0.000    |
| Qaux[5]                     | un1_dataa_int_NE:Y | Data  | 3           | 90         | 0.000      | 0.000    |
```

Figure 25 · SmartPower Activity and Hazards Report

Scenario Power Report

The scenario power report enables you to select a previously defined scenario and calculate the average power consumption and the battery life for this scenario.

To generate a scenario power report:

Note: In order to generate a scenario power report, your design must contain one or more scenarios.

1. From the Designer **Tools** menu, choose **Reports > Power > Power Scenario**. The Scenario Power Report dialog box appears.

2. Select the options you want to include in the report, and then click **OK**. The scenario power report appears in a separate window.

You can also generate the scenario power report from within SmartPower. From the **Tools** menu, choose **Reports** > **Scenario Power Report**, or click the **Scenario Power Report** button to open the Power Scenarios dialog box. By default, the report includes global design information and power sequencer summary. Specify which results you want to display by checking the boxes to be included in the report.

The power report dialog box is organized in the following panels: General, Operating Conditions, Options, and Battery Life.

General

The general panel enables you to select what to include in the report, the report format, and the scenario you want to generate the report for.

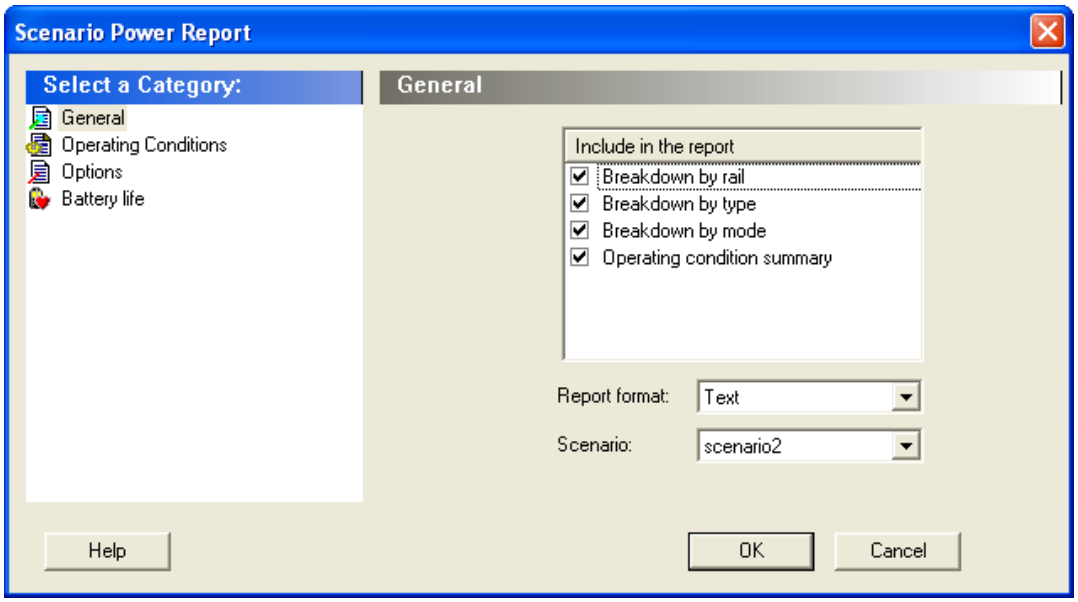


Figure 26 · SmartPower Scenario Power Report Dialog Box - General Panel

Include in the Report

Breakdown by Rail - This section shows the power consumption of each rail.

Breakdown by Type - This section enables reporting on the power consumption according to: gates, nets, clocks, core static, IOs, and memories.

Breakdown by Mode - This section enables reporting on the power consumption by mode.

Operating Condition Summary - This section reports the operating conditions.

Report Format

Select **Text** or **CSV** (Comma Separated Value) as the desired export format.

Scenario

Select a previously defined scenario to generate the report from.

Operating Conditions

The Operating Conditions panel enables you to select the operating conditions case for the current design.

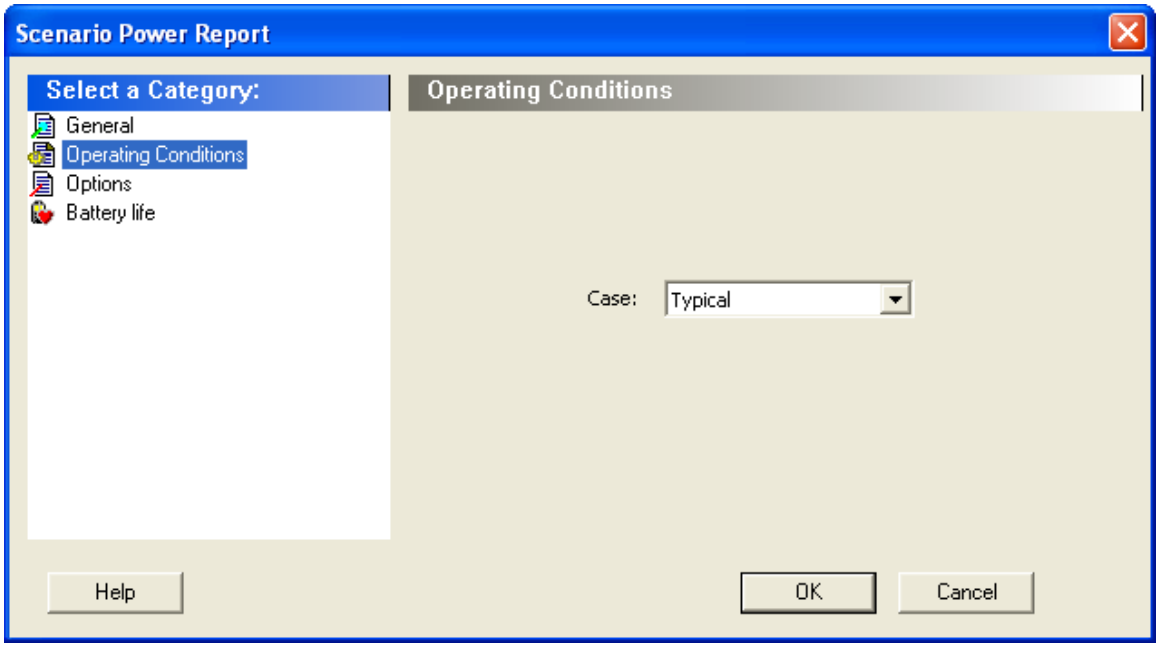


Figure 27 · SmartPower Scenario Power Report Dialog Box - Operating Conditions Panel

Options

The Options panel enables you to select power and frequency units and to use toggle rates.

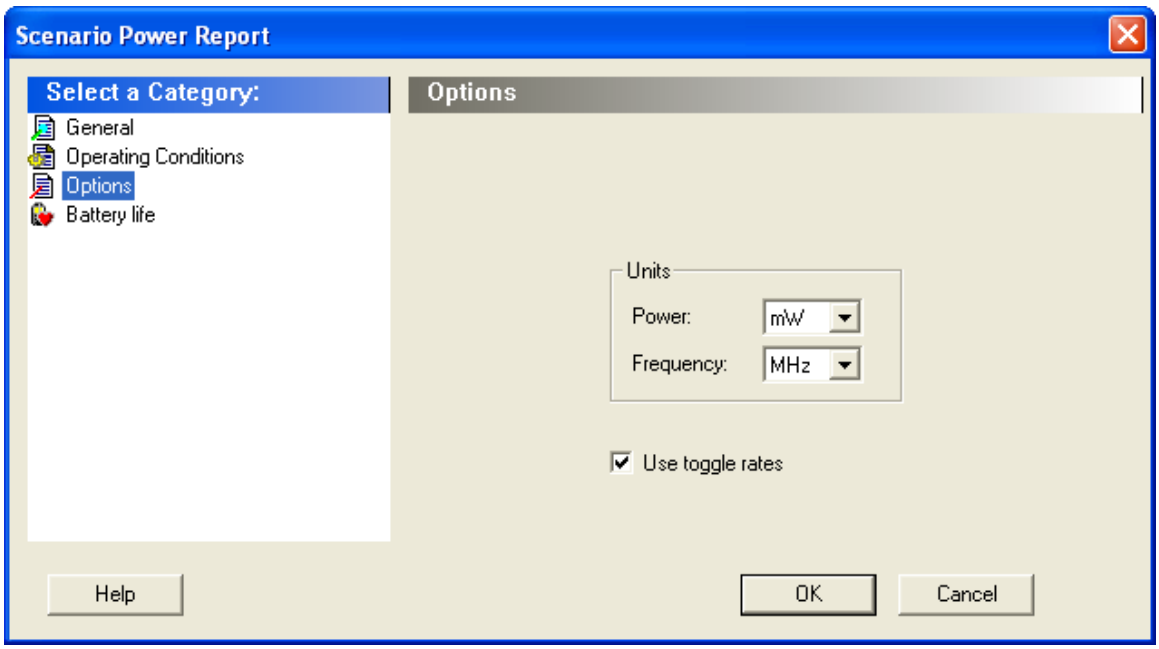


Figure 28 · SmartPower Scenario Power Report Dialog Box - Options Panel

Units

Frequency: Sets unit preferences for frequency - Hz, KHz, MHz.

Power Units: Sets unit preferences for power - W, mW, or υW.

Use Toggle Rates

When toggle rates are active (**Use Toggle Rates** box is checked), the data frequency of all the clock domains is defined as a function of the clock frequency. This updates the data frequency automatically when you update the clock frequency. Toggle rates enable you to specify the data frequency as a percentage of clock frequency, but you can no longer specify the data frequency as a number, only as a percentage of the clock frequency. To set the data frequency again, clear the **Use Toggle Rates** option.

Battery Life

The Battery Life panel enables reporting of the battery capacity and the battery life. Enter a battery capacity in MA/Hrs.

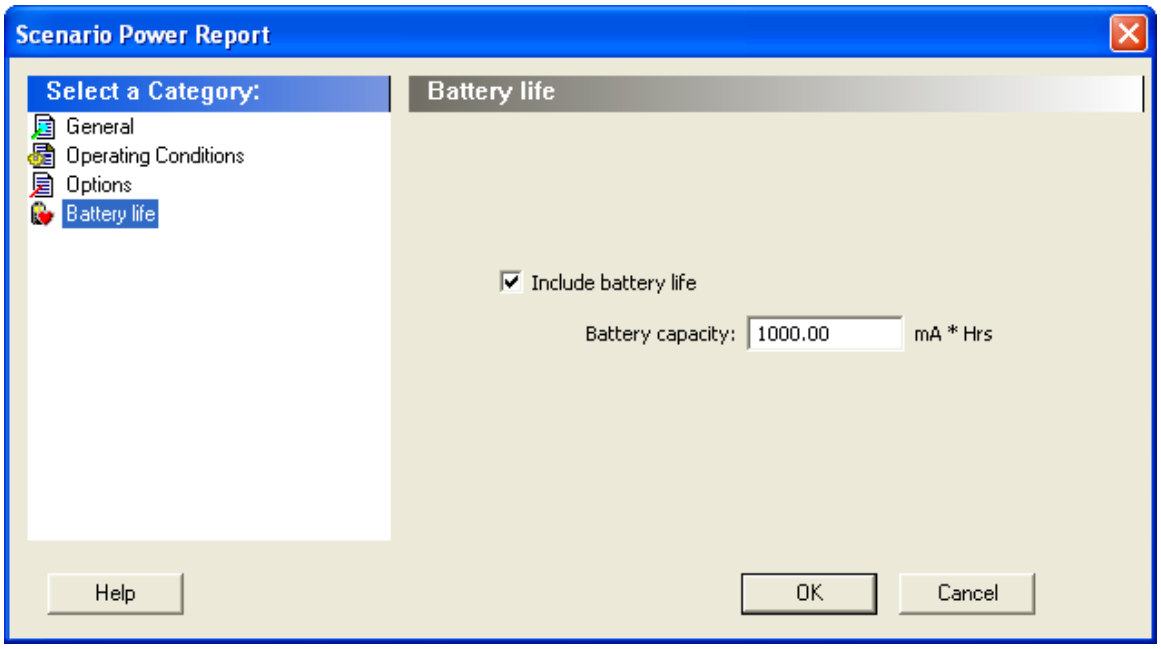
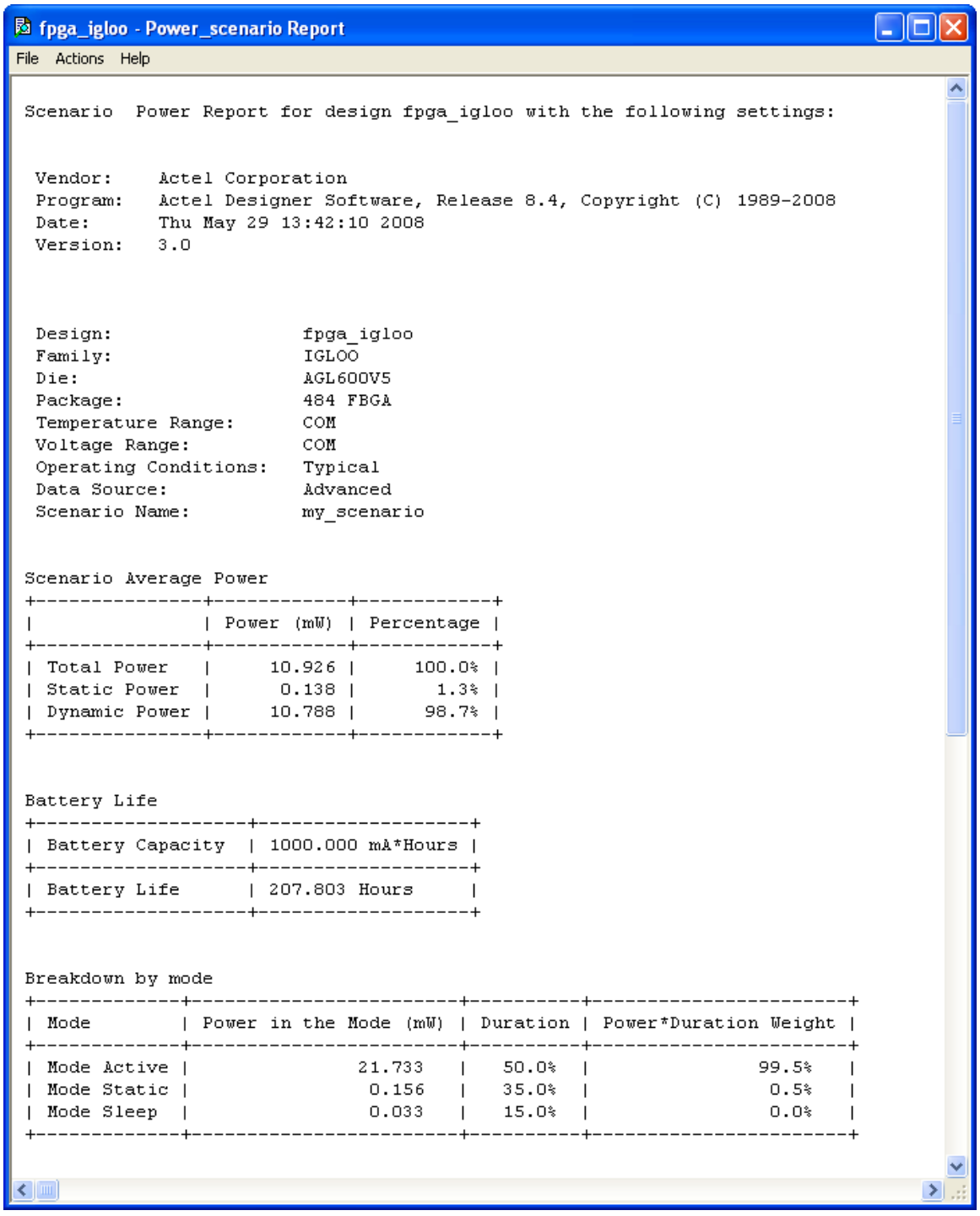


Figure 29 · SmartPower Scenario Power Report Dialog Box - Battery Life Panel

The SmartPower scenario power report returns the average power consumption and battery life for this sequence.



fpga_igloo - Power_scenario Report

File Actions Help

```
Scenario  Power Report for design fpga_igloo with the following settings:


 Vendor:    Actel Corporation
 Program:   Actel Designer Software, Release 8.4, Copyright (C) 1989-2008
 Date:      Thu May 29 13:42:10 2008
 Version:   3.0



 Design:                 fpga_igloo
 Family:                 IGLOO
 Die:                    AGL600V5
 Package:                484 FBGA
 Temperature Range:      COM
 Voltage Range:          COM
 Operating Conditions:   Typical
 Data Source:            Advanced
 Scenario Name:          my_scenario


Scenario Average Power
+---------------+------------+------------+
|               | Power (mW) | Percentage |
+---------------+------------+------------+
| Total Power   |     10.926 |     100.0% |
| Static Power  |      0.138 |       1.3% |
| Dynamic Power |     10.788 |      98.7% |
+---------------+------------+------------+


Battery Life
+-------------------+-------------------+
| Battery Capacity  | 1000.000 mA*Hours |
+-------------------+-------------------+
| Battery Life      | 207.803 Hours     |
+-------------------+-------------------+


Breakdown by mode
+-------------+------------------------+----------+-----------------------+
| Mode        | Power in the Mode (mW) | Duration | Power*Duration Weight |
+-------------+------------------------+----------+-----------------------+
| Mode Active |                 21.733 |   50.0%  |                 99.5% |
| Mode Static |                  0.156 |   35.0%  |                  0.5% |
| Mode Sleep  |                  0.033 |   15.0%  |                  0.0% |
+-------------+------------------------+----------+-----------------------+
```

Figure 30 · SmartPower Scenario Power Report

In addition to the information selected on the **Scenario Power Reports** dialog box, the report contains global design information, a mode summary and the sequence average power.

Global design information: This section shows the target family, the package and the die. It also shows information about the operating conditions, speed grade and power mode. This option is set by default.

Power Summary: This section reports the power consumption of the sequence by mode. This option is set by default.

Sequence Average Power: This section reports the average power consumption of the sequence. This option is set by default.

See Also

Report (Sequencer)

CCC Configuration Report

The Fusion Dynamic CCC (DYNCCC) and ProASIC3E Dynamic CCC prints out all the values of the configuration pins in a report. You can use these to specify the bitstream that can be shifted in via the shift register.

To view the CCC Configuration report, from the Tools menu, choose Reports > Global > CCC_configuration.

The CCC Configuration report for Fusion has some signal names that do not appear in the ProASIC3E report. A sample Fusion DYNCCC report is shown below:

```
**********************************************************************
                        Dynamic Stream Data
**********************************************************************

Product: Designer
Release: 7.2
Version: 7.2.0.13
Date   : Thu Apr 13 14:48:21 2006
Design Name: AFS600_dynpll_rc_a100_ybycout  Family: Fusion  Die: AFS600  Package: Fully
Bonded Package
Location: MIDDLE_LEFT
Instance: Core                          DEF Name: DYNCCC
          Core/U_P0  (1,40)
          Core/U_PLL  (2,40)
          Core/U_GLB  (1,41)
          Core/U_GLC  (2,41)
          Core/U_P4  (1,42)
---------------------------------------------
NAME        SDIN        VALUE       TYPE
---------------------------------------------
FINDIV      [ 6: 0]     0010011     EDIT
FBDIV       [13: 7]     0001011     EDIT
OADIV       [18:14]     00011       EDIT
OBDIV       [23:19]     00001       EDIT
OCDIV       [28:24]     00000       EDIT
```

```
OAMUX       [31:29]     101         EDIT
OBMUX       [34:32]     111         EDIT
OCMUX       [37:35]     110         EDIT
FBSEL       [39:38]     10          EDIT
FBDLY       [44:40]     10100       EDIT
XDLYSEL     [45]        1           EDIT
DLYGLA      [50:46]     01001       EDIT
DLYGLB      [55:51]     00000       EDIT
DLYGLC      [60:56]     00000       EDIT
DLYYB       [65:61]     10001       EDIT
DLYYC       [70:66]     11100       EDIT
STATASEL    [71]        1           MASKED
STATBSEL    [72]        1           MASKED
STATCSEL    [73]        1           MASKED
VCOSEL      [76:74]     010         EDIT
DYNASEL     [77]        0           MASKED
DYNBSEL     [78]        1           MASKED
DYNCSEL     [79]        1           MASKED
RESETEN     [80]        1           READONLY
RXASEL      [81]        0           MASKED
RXBSEL      [82]        0           MASKED
RXCSEL      [83]        0           MASKED
OADIVHALF   [84]        0           EDIT
OBDIVHALF   [85]        0           EDIT
OCDIVHALF   [86]        0           EDIT
GLMUXCFG    [88:87]     00          MASKED
```

A sample ProASIC3E DYNCCC report is shown below.

```
**********************************************************************
                        Dynamic Stream Data
**********************************************************************

Product: Designer
Release: 7.2
Version: 7.2.0.0
Date   : Tue May 02 15:50:01 2006
Design Name: dynccc  Family: Proasic3E  Die: A3PE600  Package: Fully Bonded Package
Location: MIDDLE_RIGHT
Instance: I_dynccc                    DEF Name: DYNCCC
          I_dynccc/U_DYN  (196,40)
          I_dynccc/U_PLL  (195,40)
          I_dynccc/U_GLB  (196,41)
          I_dynccc/U_GLC  (195,41)
---------------------------------------------
NAME        SDIN        VALUE       TYPE
---------------------------------------------
FINDIV      [ 6: 0]     1100101     EDIT
FBDIV       [13: 7]     0000110     EDIT
```

OADIV	[18:14]	00101	EDIT
OBDIV	[23:19]	00101	EDIT
OCDIV	[28:24]	00101	EDIT
OAMUX	[31:29]	101	EDIT
OBMUX	[34:32]	101	EDIT
OCMUX	[37:35]	101	EDIT
FBSEL	[39:38]	00	EDIT
FBDLY	[44:40]	00110	EDIT
XDLYSEL	[45]	0	EDIT
DLYGLA	[50:46]	00101	EDIT
DLYGLB	[55:51]	10001	EDIT
DLYGLC	[60:56]	10001	EDIT
DLYYB	[65:61]	00101	EDIT
DLYYC	[70:66]	00101	EDIT
STATASEL	[71]	1	MASKED
STATBSEL	[72]	1	MASKED
STATCSEL	[73]	1	MASKED
VCOSEL	[76:74]	000	EDIT
DYNASEL	[77]	0	MASKED
DYNBSEL	[78]	0	MASKED
DYNCSEL	[79]	0	MASKED
RESETEN	[80]	1	READONLY

Report (Global Usage)

Creates a report containing information about the net(s) that are assigned or routed using Global or LocalClock resources.

```
report -type globalusage filename
```

Arguments

`-type globalusage`

Specifies the type of report to generate.

`filename`

Specifies the name and destination of the generated Global Usage report.

Supported Families

ProASIC[PLUS], ProASIC

Exceptions

- None

Examples

This example generates a Global Usage report and saves it to a file named globalusage_rpt.txt:

```
>report -type globalusage globalusage_rpt.txt
```

See Also

Global Usage report

Tcl documentation conventions

Global Usage Report

The Global Usage report provides information about the net(s) that are assigned or routed using Global or LocalClock resources.

You can generate this report in either pre or post route state of the design.

In pre-routed state, the Global Usage report displays the following information:

- Net(s) assigned to Global resources
- Net(s) assigned to LocalClock resources

In post-routed state, the Global Usage report displays the following information:

- Net(s) that are routed using Global resources
- Net(s) that are routed using LocalClock resources

- Net(s) that are constrained to use LocalClock resources but routed using Regular resources. In this case, the tool displays a warning message at the end of the report.

To create a Global Usage report:

From the **Tools** menu, choose Reports > Global > GlobalUsage.

See Also

report (Global Usage) Tcl command

Designer Block Report

The Designer Block report is available in **Tools > Reports > Block**. It includes a compile, global, datasheet, and interface report and Designer Block-related information.

The block reports are available in the Project Manager Files tab after you generate your block and instantiate it in your project. Double-click the report to view the contents.

In the Project Manager there is a header_report.log that contains only the options used to generate the block. This information is available in each report you generate from the Designer > Tools menu.

Compile

Use it to evaluate resources and manage the globals in the other blocks and the <top> design (if necessary).

Datasheet

Lists block timing and I/O placement information. If the block is preserved during instantiation (both placement and routing) you can expect to get the same results as are shown in this report.

Global

Lists global usage in the block. Useful if you want to evaluate the globals used by the block / manage globals in the overall design.

Interface

The Interface section lists:

- Connection information for interface macros connected to the block ports
- Block placement information, including each port and its fanout, type (pad, clock, global, etc.), direction, and name
- Information on legal move locations, useful if you are instantiating multiple blocks in one design

Compile Report

The Compile Report appears in the Designer Log window after compile is complete. It is divided into the following sections:

Parameters Used to Run Compile

All the information about the design, including the device selection and compile options used to run compile. Some options are not shown if you did not select them; for example, if the option demote_globals is OFF than the option demote_globals_max_fanout does not appear in the report.

Warnings, Errors, and the Netlist Optimization Report

Lists any warnings or errors encountered during compile. Also contains the Netlist Optimization report that lists out the optimized macros (deleted blocks) in your design.

Reading User PDC (Physical Design Constraints) File(s) Postcompile

Lists out any PDC related errors and warnings encountered during compile.

Compile Report - Device Utilization Report

The Device Utilization Report includes the following:

- Complete device utilization: Summary for all the resources used in the final optimized netlist.
- Global Information: Describes the number of chip and quadrant clocks in the design and the device.
- Core Information: Describes the total number of macros in the netlist and how many tiles they are using.
- I/O function: Detailed information about I/O's, such as how many Diff I/Os exist in the netlist, etc.
- I/O Technology: Summary of I/O technology used in the netlist.
- I/O Bank Resource Usage: Summarizes each I/O bank, including details on voltages, I/O pairs, Vref I/Os, Vref Pins, etc.
- I/O Voltage Usage: Lists I/Os by voltage used in the netlist and device resources available for each voltage (using the iobank and placement information).

Net Information Report

The section may not appear if there is no net to report. The Net Information Report includes the following sections:

- List of nets that drive enable flip-flops that have been remapped to a 2-tile implementation
- List of nets that have been assigned to a chip global resource
- List of nets that have been assigned to a quadrant global resource
- List of nets that nets have been assigned to a LocalClock resource using PDC constraints
- High fanout nets in the post-compile netlist

- Nets that are candidates for clock assignment and the resulting fanout

Note: If the driver macro of a clock net is fixed in a quadrant clock location then this net will show in the quadrant clock net report.

The number of clock nets (chip+quadrant) can be less than the number reported in the device utilization (such as in the case of PLL using a YB and not the GLB).

Net Information Report Types

INT_NET - Internal nets

CLK_NET - More than 75% of pins driven by this net are clock pins

SET/RESET_NET - More than 75% of pins driven by this net are set or reset pins

Net Information Source Types:

NETLIST - Clock from the netlist

PDC PROMOTED - Promoted because of a PDC constraint

AUTO PROMOTED - Promoted automatically by compile; change the compile options if you do not want to promote this net.

ESSENTIAL - Global clock from the netlist that cannot be demoted (such as the PLL or CLKBIBUF).

Net Information Report Region (Definition)

The region is the clock region for the local and quadrant clocks. For example,
Local clock regions chip_T1, chip_T2:B5, quandrant_T1

The quadrant clock regions list is as follows: quadrant_UL, quadrant_UR, quadrant_LL, quadrant_LR

Block Information Report

This section lists the name of the module, the name of the instance, the number of macros and nets used in your block(s), and information on how conflicts between blocks were resolved (if any).

Exporting Files

Designer supports different types of files to export.

Supported File Types

The following table shows a complete list of files that you can export along with the supported family.

Note: Designer does not support exporting VHDL 87 format.

Files	File Extension	Export Type	Family
Actel Flattened Netlist	*.afl	Netlist file	All
Actel Internal Netlist	*.adl	Netlist file	All
Standard Delay Format	*.sdf	Timing file	All
STAMP	*.mod, *.data	Timing file	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A
Tcl script file	*.tcl	Script file	All
Verilog Netlist	*.v	Netlist file	All
VHDL Netlist	*.vhd	Netlist file	All
EDIF Netlist file	*.edn	Netlist file	All
Log File	*.log	Log file	All
STAPL	*.stp	Programming file	IGLOO, Fusion, and ProASIC3, ProASICPLUS
PDB	*.pdb	Programming file	IGLOO, Fusion, and ProASIC3
SVF	*.svf	Programming file	IGLOO, Fusion, and ProASIC3

Files	File Extension	Export Type	Family
ISC 1532	*.bsd, *.isc	Programming file (*.bsd) ISC Data file (*.isc)	IGLOO, Fusion, and ProASIC3
Bitstream	*.bit	Programming file	Fusion, IGLOO, ProASIC3, ProASIC, ProASICPLUS
Programming file (legacy)	*.fus	Programming file	ACT1, ACT2, ACT3, MX, 1200XL, 3200DX
Actel programming file	*.afm	Programming file	ACT1, ACT2, ACT3, MX, 1200XL, 3200DX, SX, SX-A, eX, Axcelerator
Location constraint file	*.loc	Other file	SX-A
Routing Segmentation file	*.seg	Other file	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator
Silicon Explorer Probe file	*.prb	Debugging file	ACT1, ACT2, ACT3, MX, 1200XL, DX, SX, SX-A, eX, Axcelerator
ProASIC Constraints file	*.gcf	Constraints file	ProASIC
ProASICPLUS Constraints file	*.gcf	Constraints file	ProASIC PLUS(Timing constraints in GCF are not supported)
Combiner Info	*.cob	Other file	ACT1, ACT2, ACT3, MX, 1200XL, DX, SX, SX-A, eX, Axcelerator, IGLOO, ProASIC3
BSDL file	*.bsd	Debugging file	IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, MX, eX, and SX/SX-A
Criticality	*.crt	Constraints file	ACT1, ACT2, ACT3, MX, 1200XL, DX
I/O buffer information specification (IBIS)	*.ibs	Other file	MX, ProASIC^{PLUS}, IGLOO, ProASIC3; the IBIS file provides a standard file format for recording parameters like driver output impedance, rise/fall time, and input loading, which may then be used by any software application.
PIN	*.pin	Constraints file	ACT1, ACT2, ACT3, MX, 1200XL,

Files	File Extension	Export Type	Family
			DX, SX, SX-A, eX
SDC	*.sdc	Constraints file	SX-A, eX, Axcelerator, ProASICPLUS, IGLOO, ProASIC3, Fusion
Block Files	*.v; *.vhd; *.cxf; *.cdb	Netlist file (*.v; *.vhd), info file for Libero IDE (*.cxf), and Designer block file (*.cdb)	IGLOO, Fusion, ProASIC3 and Axcelerator
Physical Design Constraint	*.pdc	Constraints file	IGLOO, Fusion, ProASIC3 and Axcelerator
Design Constraint file	*.dcf	Constraints file	ACT1, ACT2, ACT3, MX, 1200XL, DX, SX, SX-A, eX

To export a file:

1. From the File menu, choose Export and then select the type of file you wish to export.

2. Specify file name and file type and click OK.

Note: You must compile your design before you export an SDC file.

To export a GCF file:

1. From the File menu, choose Export Constraint Files.

2. Specify the file name and click OK.

3. Select the type of information you want to export:

Pin locations: Select to export the I/O placement and region constraints related to the I/Os only.

Placement constraints: Select to export all placement constraints, including I/O and core constraints.

All GCF constraints: Select to export all constraint information.

4. Click **OK**.

To export a PDC file:

1. From the File menu, choose Export Constraint Files.

2. Specify the file name and select a directory for it. Click Save. The Export Physical Design Constraints (PDC) dialog box appears.

3. Select the type of information you want to export:

Pin locations and attributes: Select to export information about the pin locations and attributes only.

Placement constraints: Select to export all user-defined constraints.

Complete placement information: Select to export all the information about the I/O locations, I/O attributes, and logic placement.

4. Click OK.

A message appears in the Log window telling you if the Export command succeeded.

Note: The content of the exported PDC file depends on the state of your design.

Pre-Compile: The exported PDC file will contain constraints that are stored in the database from the last valid compile state. Export may fail if you did not run Compile at least once before running Export.

Post-Compile: The exported PDC file will contain constraints that are currently stored in the database.

See Also

Exporting STAMP files

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. From the **File** menu, choose **Save** or click the save icon in the toolbar.
2. Enter the File name and click **Save**. The default file name is the name you previously entered in the setup dialog box. The default format is ADB. Make sure you save your file in 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, select **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 informations is not saved and the <design_name>.adb file remains unchanged.

Add or Edit Profile Dialog Box

Use the Add or Edit profile dialog box to add a new tool or edit an existing tool profile.

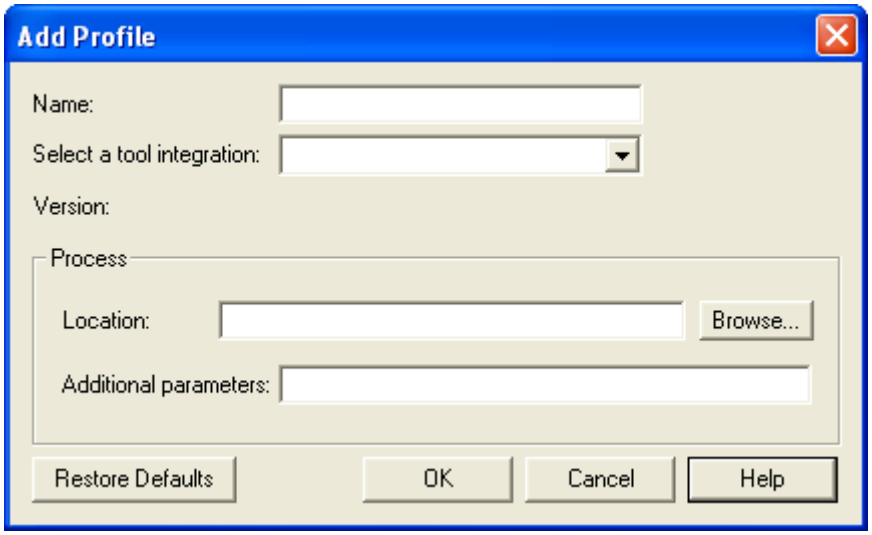


Figure 31 · Add Profile Dialog Box

Name

Enter the name for the tool. The name appears in the Project Profile dialog box.

Select a tool integration

Select a tool from the drop down list of integrated tools.

Version

Lists your version of the selected tool.

Process:

- **Location**: Enter the location of the tool, or click Browse and locate the project directory for the .exe file.
- **AdditionalParameters**:Enter additional arguments you want passed to the tool.

To access this dialog, from the **Profile** dialog box, click **Add** or **Edit**.

See Also

Project Profile dialog box

Project profile

Add Cores to Vault Dialog Box

This dialog box (shown below) enables you to download cores from a web repository into a Vault.

A Vault is a local directory (either local to your machine or on the local network) that contains cores downloaded from one or more repositories. A repository is a location on the web that contains cores that can be included in your design.

The Catalog displays all the cores in your Vault.

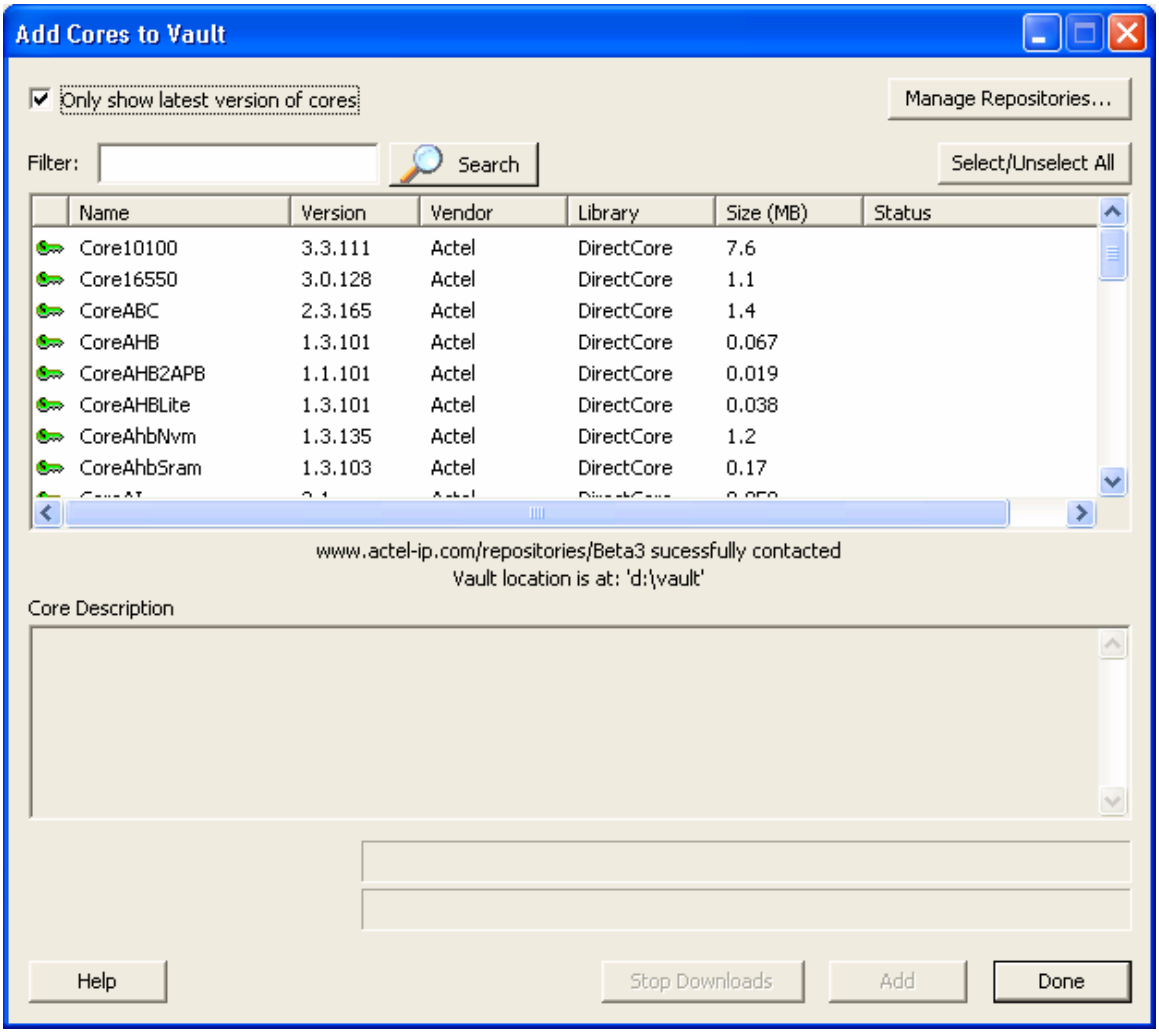


Figure 32 · Add Cores to Vault Dialog Box

Only show latest version of cores is checked by default. This option, if checked, shows the latest versions of cores that are not in the Vault , and also filters out any duplicate cores that have the same Vendor, Library, and Name, with an earlier version number.

If the checkbox is de-selected the dialog box displays all cores, including those already loaded into the Vault. The status column indicates if a core has already been loaded.

Use the Filter to find any string that exists either in the core name or the Core Description. By default the filter contains a beginning and ending '*', so if you type 'controller' you get all cores with controller in the core name (case insensitive search) or in the core description.

The colored icons indicate the license status. Blank means that the core is not license protected in any way. Colored icons means that the core is license protected, with the following meanings:

Green Key - Fully licensed; supports the entire design flow.

Yellow Key - Has a limited or evaluation license only. You can purchase a license to generate an obfuscated or RTL netlist.

Yellow Key with Red Circle - License is protected; you are not licensed to use this core.

Stop Downloads - Interrupts the download for any cores being added to your Vault.

Catalog Display Options Dialog Box

The Catalog Display Options dialog box enables you to customize your Catalog display. You can change the way cores are listed, the default sort order, show/hide the filter fields, and show/hide the core version. To display this dialog box, right click the Name column heading and choose Catalog Display Options.

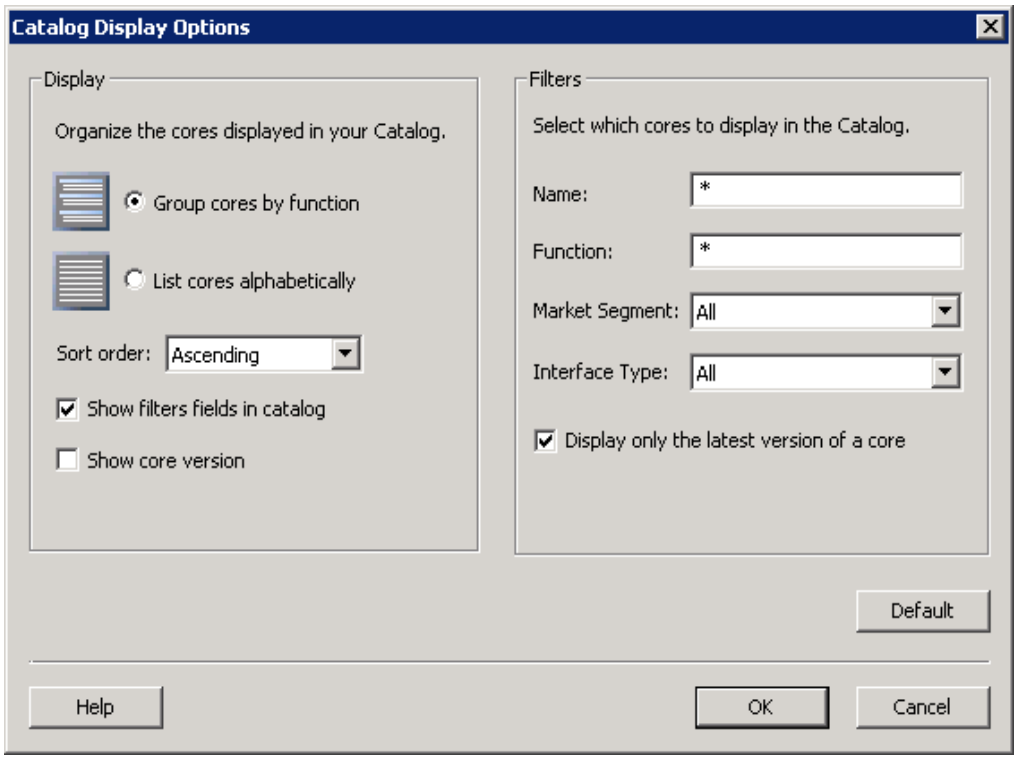


Figure 33 · Catalog Display Options Dialog Box

Display

Group cores by function - Displays a list of cores, sorted by function. Click any function to expand the list and view specific cores.

List cores alphabetically - Displays an expanded list of all cores, sorted alphabetically. Double click a core to configure it. This view is often the best option if you are using the filters to customize your display.

Sort order - Sets your Name sort order to Ascending or Descending. Click the Name column heading to change this option without opening the Display Options dialog box.

Show filter fields in Catalog - Shows/hides the Name and Function filter fields.

Show core version - Shows/hides the core version. The core version appears adjacent to the Name column, but you cannot sort according to version.

Filters

Name - Type text in the Name field to filter cores by name. Note that the Catalog does not display results until you type an entire name (such as Accumulator, Ripple), or you type the beginning letters of the name and a wildcard, such as: accu*

Function - Type text in the Function field to filter cores by function.

Market Segment - Select a market segment to narrow your search. You can specify a market segment for published cores in your repository.

Interface Type - Specify an interface type to narrow your search. Actel offers a range of AMBA enabled cores that allow you to design an AMBA-based system. Select AMBA or Not AMBA to refine your search in the catalog.

Display only latest version of a core - Shows/hides older versions of cores; this feature is useful if you are designing with an older family and wish to use an older core.

Default resets the options to their original settings.

CDB File Organization Dialog Box

The CDB File Organization dialog box manages conflict resolution between the blocks in your design.

It is important to preserve the order of your CDB files if they are interdependent. Use the CDB File Organization dialog box to set the compile order for your CDB files (as shown in the figure below). CDB file organization is a pre-module setting; modifying the CDB order will apply to all ADBs of a given module.

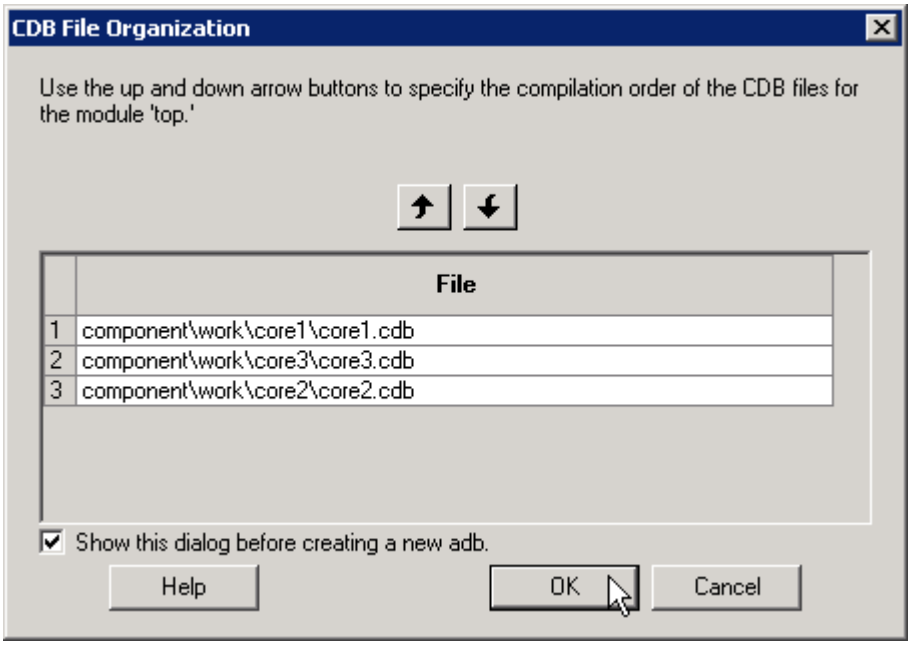


Figure 34 · CDB File Organization Dialog Box

Use the up and down arrows to specify the order, or drag the files into order. Select the checkbox to show the dialog before you create a new ADB file.

Change All Links Dialog Box

The Change All Links dialog box enables you to update/change all the links for the files in your project at once.

This dialog box is useful when you are linking to shared network folders that may be renamed by other users, or to folders on your local machine that have been renamed.

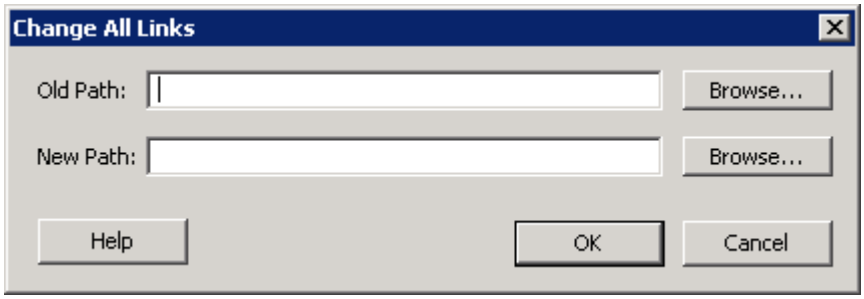


Figure 35 · Change All Links Dialog Box

Old Path - Location of your old project files. Click the browse button to navigate to the location.

New Path - Location of your new project files. Click the browse button to navigate to the location.

Configure Flow Dialog Box

Physical synthesis with PALACE is available only for IGLOO, ProASIC3, Axcelerator, and ProASIC^{PLUS} devices.

When you click Configure Design Flow in the Project Manager Design Flow window you can choose to configure the flow and to use PALACE and enable physical synthesis (as shown below).

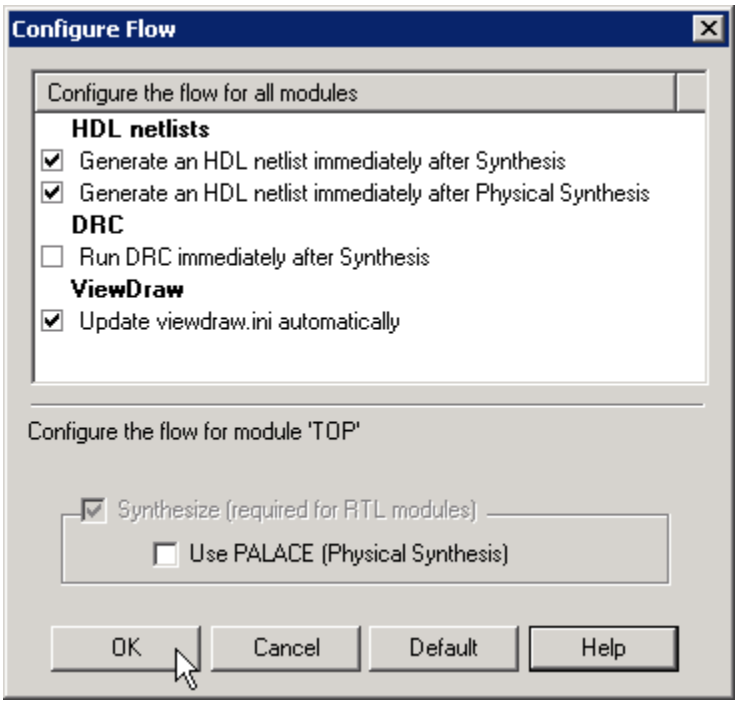


Figure 36 · Configure Flow Dialog Box

Configure the Flow for all Modules

The HDL project level options enable you to specify whether or not you want to generate your HDL netlists after synthesis.

HDL Netlists

- **Generate an HDL netlist immediately after synthesis** - Select this option if you run post-synthesis simulation. If you do not run post-synthesis simulation, de-select this option to start Designer more quickly.
- **Generate an HDL netlist immediately after physical synthesis** - Select this option if you run physical simulation. If you do not run post physical synthesis simulation, de-select this option to start Designer more quickly.

DRC

Run DRC immediately after synthesis - This option evaluates the netlist generated by Synplify to ensure it does not have any connection problems.

ViewDraw

Update viewdraw.ini automatically - May be useful if the Project Manager does not create a valid viewdraw.ini file. Click the checkbox to enable.

File Detection - Detect new files on disk automatically enables the Project Manager to see new files when you add them to your project. If you deselect this option, you must add the new files manually.

SmartDesign core generation - Generate resource report creates and saves the SmartDesign resource report after you generate a core.

FlashPro options - Input programming file for FlashPro sets your input programming file. PDB files enable you to configure the security settings in FlashROM and Flash Memory from FlashPro.

Configure the Flow for Module <module_name>

You can also choose to enable or disable synthesis if you are using a structural implementation. Synthesis is required for RTL modules and cannot be disabled.

Enabling physical synthesis with PALACE automatically creates a new project implementation in Libero and displays the PALACE icon in the Project Manager Design Flow window. See the online help for more information on how to use the PALACE tool for physical synthesis.

To view this dialog box, click **Configure Flow** in the Project Manager Design Flow window.

Convert Project Dialog Box

The Convert Project dialog box enables you to create a backup file of your project before you convert the project data to use with your current version of Libero IDE (as shown in the figure below).

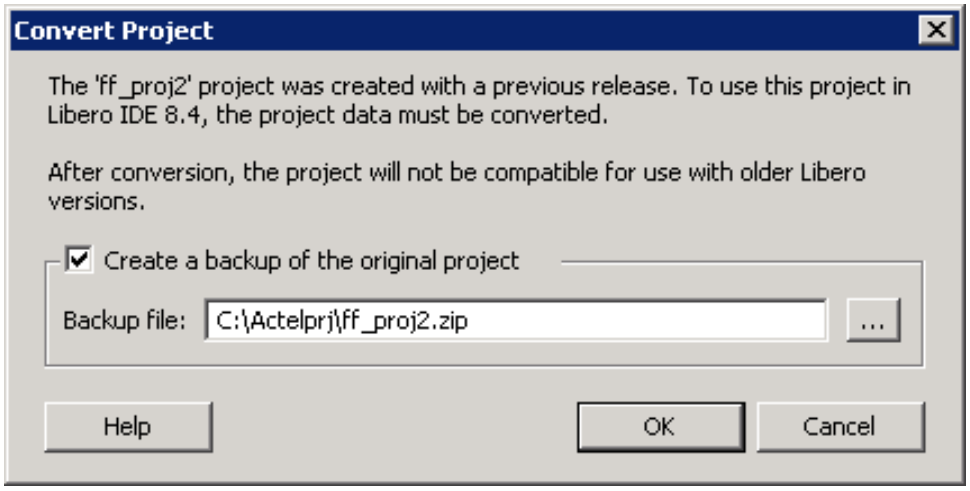


Figure 37 · Convert Project Dialog Box

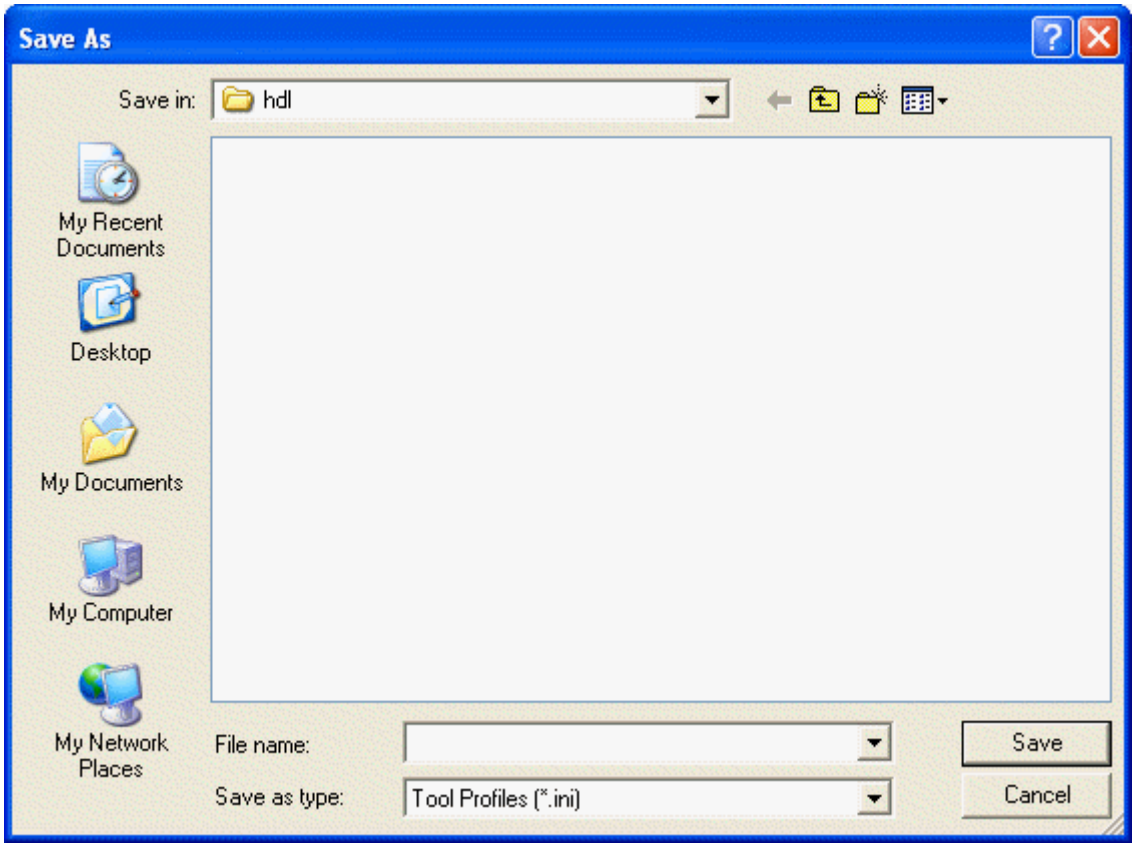
Create a backup of the original project - Creates a backup of your project and saves it as a zip file in the directory you specify. Click the browse button to set the location of the backup file.

Export Profiles Dialog Box

Use the Export Profiles dialog box to create an INI file and save your tool profiles.

After you create an INI file, you can import it into other Libero IDE projects.

Specify a filename and click **Save**, or click **Cancel** to return to the Project Manager.



Save in

Specifies the director in which to save your exported tool profile. Browse to select a different directory.

File name

Type the file name for your exported tool profile.

Save as type

Specifies the file type displayed in the dialog box. Only INI files are supported for tool profiles.

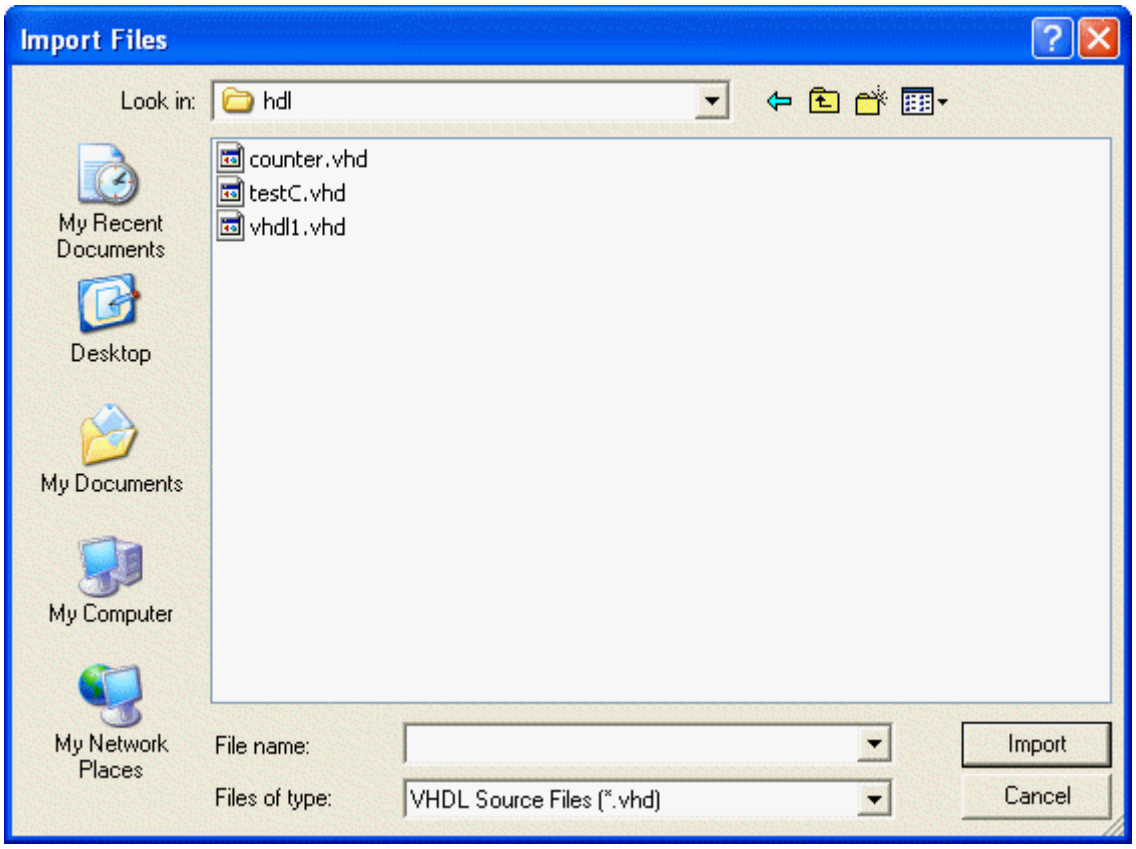
To access this dialog: from the **File** menu, choose **Export > Profiles**.

Import Files Dialog Box (Project Manager)

Use the Import Files dialog box to add new files to your project in the Libero IDE Project Manager.

You can import schematics, VHDL or Verilog source files, stimulus files, SDC, PDC, VCD, and SAIF files, SmartGen cores, and even tool profiles (from other Libero IDE projects).

Browse to and select the file you wish to add and click **Import**, or click **Cancel** to return to the Project Manager.



Look in

Specifies your current directory. Browse to find your file if it is not listed here. If you are in the correct directory and your file is not listed here, select the **File of type** extension to match it.

File name

Type the file name, or browse to its location and select it.

File of type

Specify the file type displayed in the dialog box.

To access this dialog: from the **File** menu, choose **Import Files**.

Manage Repositories Dialog Box

A repository is a location on the web that contains cores that can be included in your design.

The Manage Repositories dialog box (shown in the figure below) enables you to specify which repositories you want to display in your Vault. The Vault displays a list of cores from all your repositories, and the Catalog displays all the cores in your Vault.

The default repository cannot be permanently deleted; it is restored each time you open the Manage Repositories dialog box.

Any cores stored in the repository are listed by name in your Vault and Catalog; repository cores displayed in your Catalog can be filtered like any other core.

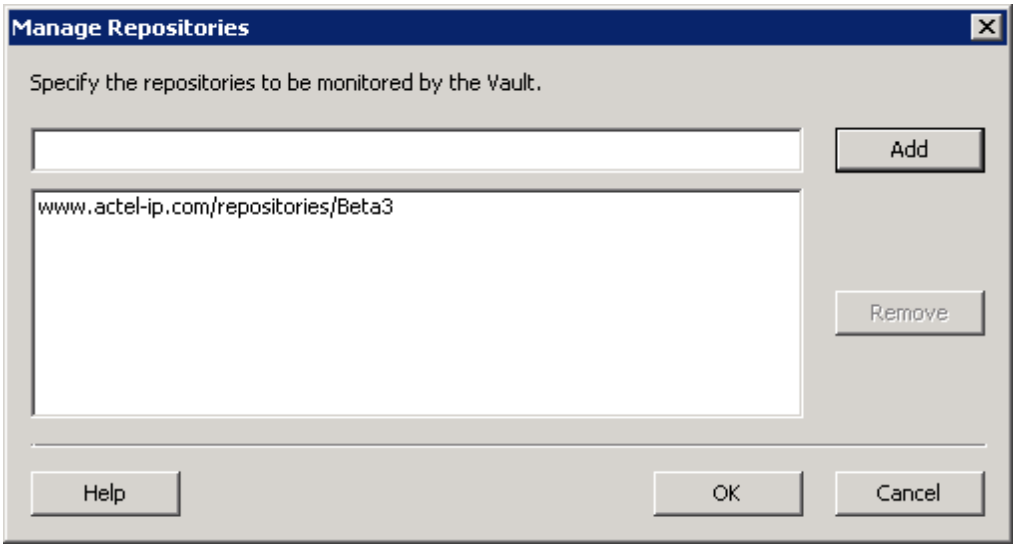
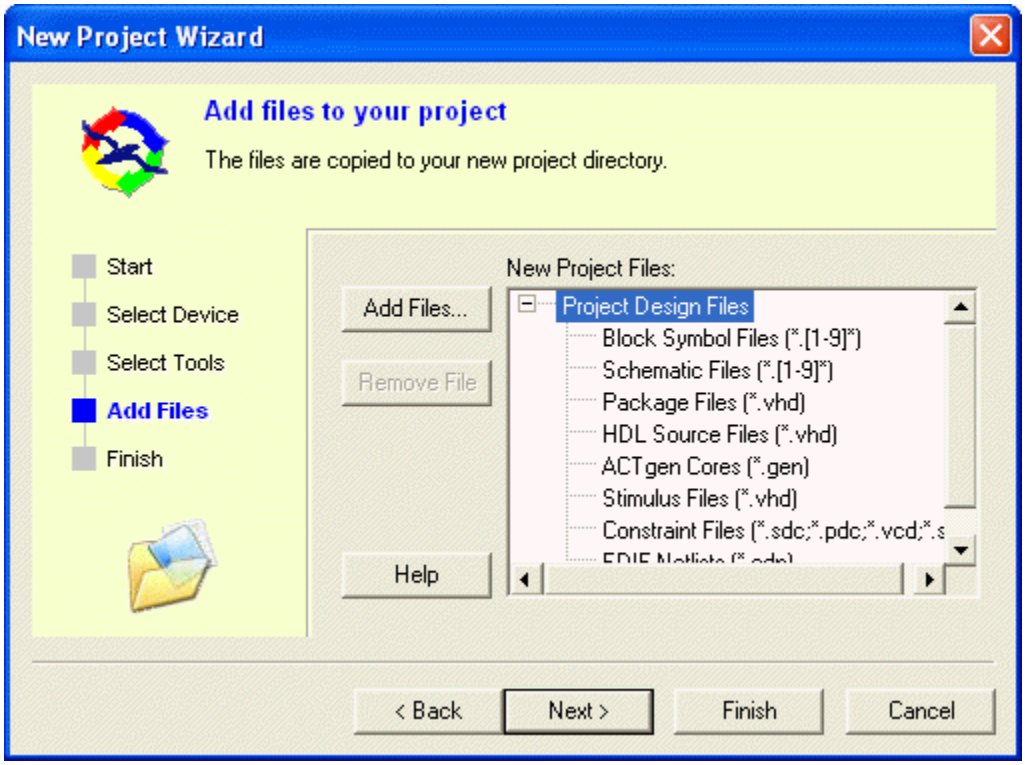


Figure 38 · Manage Repositories Dialog Box

Type in the address and click the Add button to add new repositories. Click the Remove button to remove a repository (and its contents) from your Vault and Catalog. Removing a repository from the list removes the repository contents from your Vault.

New Project Wizard: Add Files

This step of the New Project Wizard enables you to copy files to your new project directory.



Add Files

Select a file type in the New Project Files list and click Add Files to add a file to your new project.

Remove File

Select a file you have added in the New Project Files list and click Remove Files to delete it.

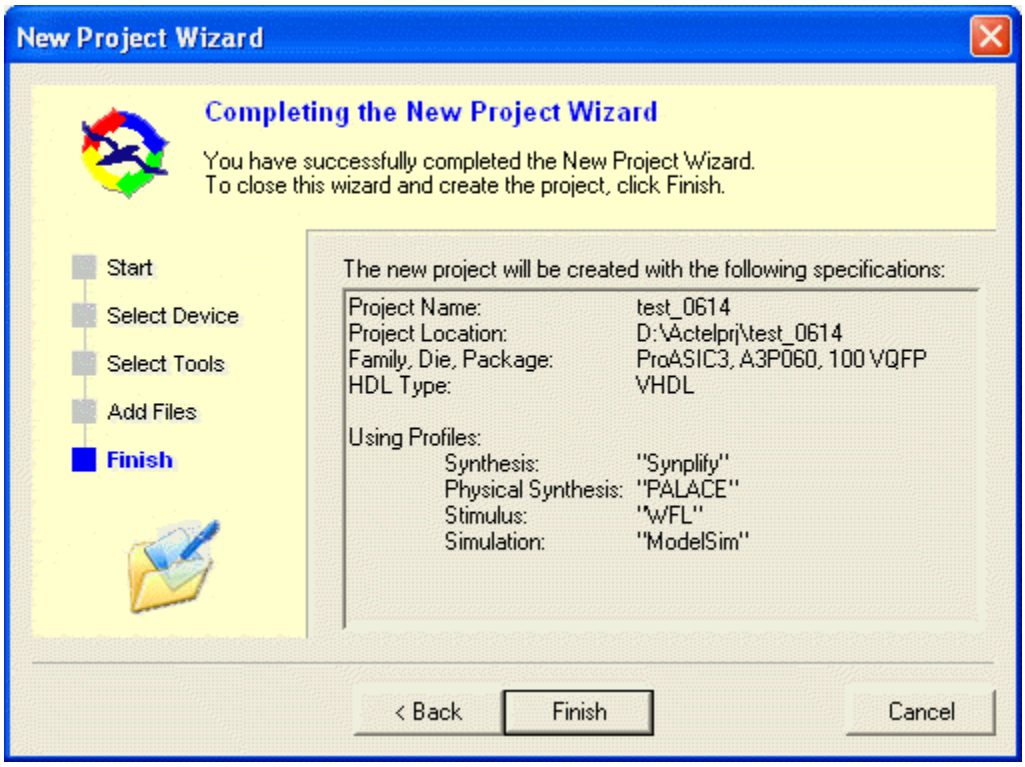
To add a file:

1. Select the file type in New Project Files.
2. Click **AddFile**.
3. Select the file to add and click **Add**. The file appears in New Project Files.
4. Continue adding files for your new project. When done, click **Next**. Drag files to re-order them in the New Project Files hierarchy.
5. Click **Next**.

To access this dialog, from the **Project** menu, choose **NewProject** and follow the instructions in the New Project Wizard.

New Project Wizard: Completing

This dialog box displays a summary of all the elements of your new project. Review the list and click **Back** to return to the previous screens and change your project options. Click **Finish** to create the project as shown.



To access this dialog, from the **Project** menu, choose **NewProject** and follow the instructions in the wizard.

New Project Wizard: Start

Use the New Project Wizard to create new projects in the Project Manager. The wizard creates all the directories for your project in one top-level project directory.

Enter information about your new project and click Next.

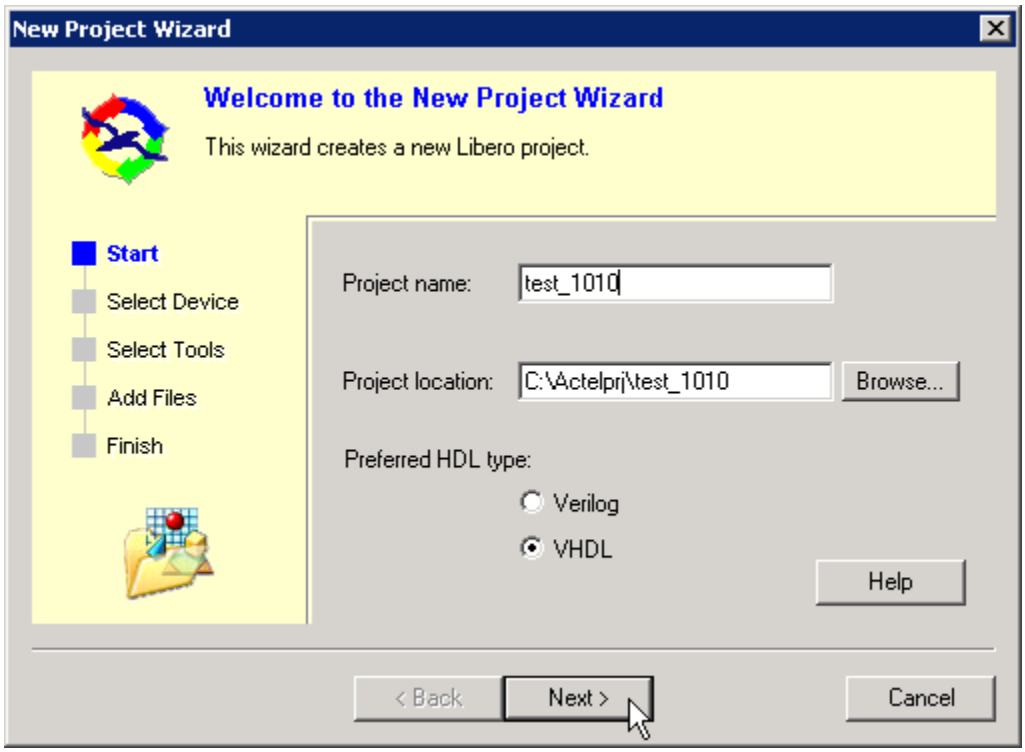


Figure 39 · New Project Wizard Dialog Box - Start

Project Name

Type the project name.

Project Location

Accept the default location or browse to the new location where you can save and store your project. All files for your project are saved in this directory.

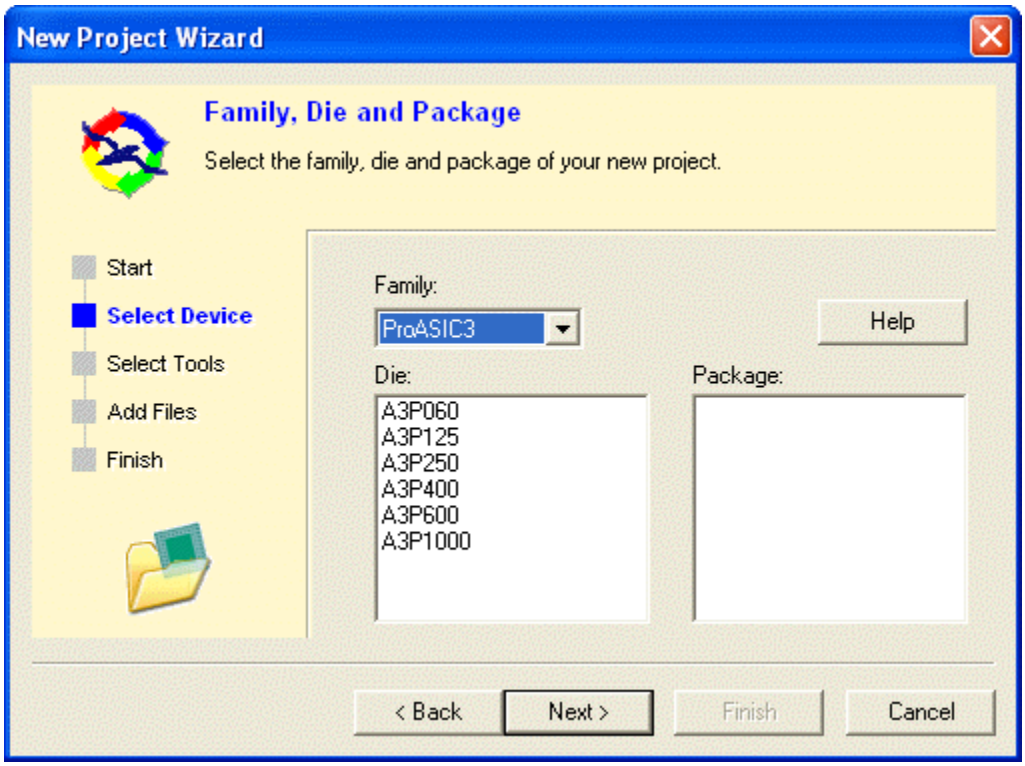
Preferred HDL type

Select Verilog or VHDL. Libero IDE supports mixed-HDL design flows.

To access this dialog, from the **Project** menu, choose **New Project**.

New Project Wizard: Select Device

Select the family device, die, and package you intend to use for your project. You may select a package only after you have selected a family and die.



Family

Select your device family from the drop-down menu.

Die

Select the die for your project.

Package

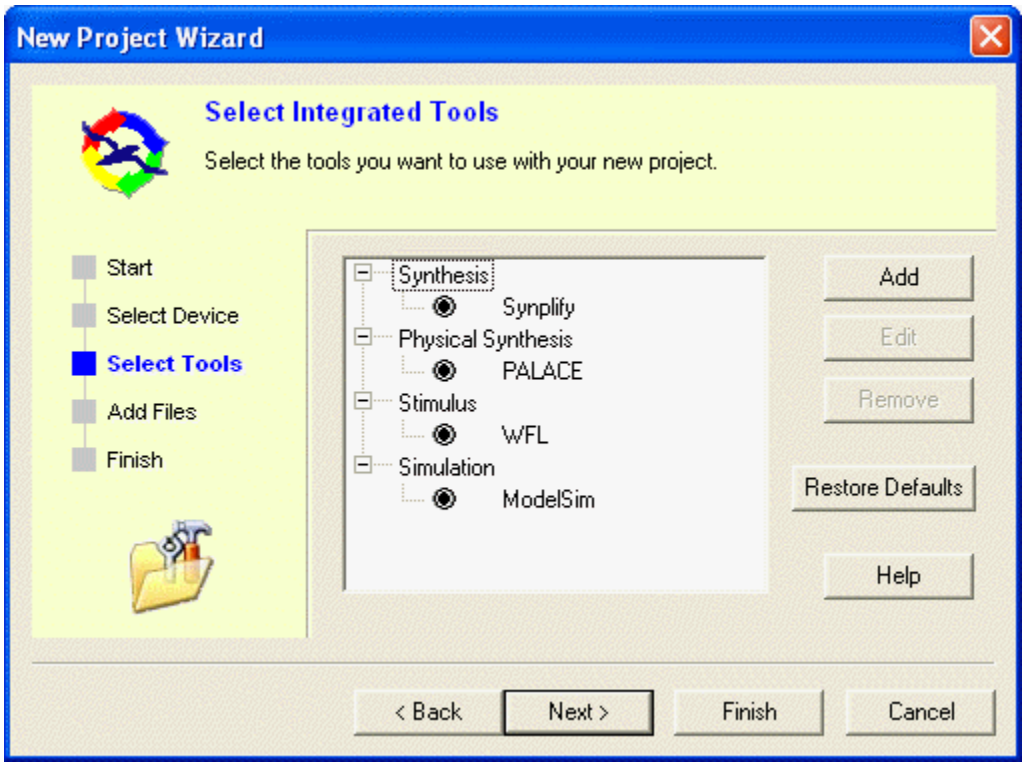
Select the package for your project. The package list varies according to the die you select.

Click Next to add files or Finish to open the new project without importing files.

To access this dialog, from the **Project** menu, choose **New Project** and follow the instructions in the New Project Wizard.

New Project Wizard: Select Integrated Tools

Select the tools you want integrated with this project. Click Add and select the tool type, or select a tool and click Edit to specify the tool location. The example shows the tools available in the default Libero IDE installation. This dialog box sets your project profile information.



Add

Click and select a tool type, and then enter the tool profile information.

Edit

Opens the tool profile manager for the tool you selected.

Remove

Select a tool and click Remove to remove a tool from the project.

Restore Defaults

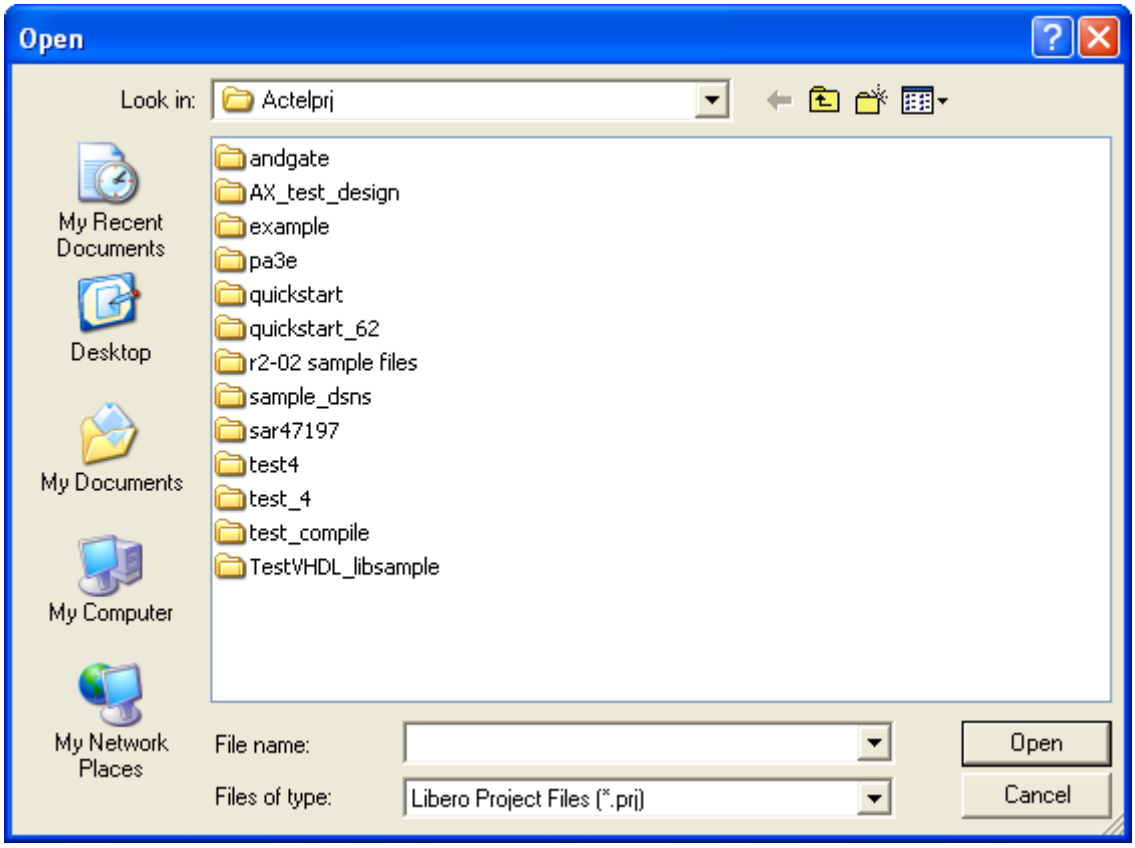
Restores the default tools and locations.

Click Next to add files or Finish to open the new project without importing files.

To access this dialog, from the **Project** menu, click **New Project** and follow the instructions in the New Project Wizard.

Open Project Dialog Box

Use the Open Project dialog box to navigate to and open existing projects in the Project Manager. Browse to your project and click **Open**, or click **Cancel** to return to the Project Manager.



Look in

Specifies the directory that contains your project.

File name

Type the file name, or browse to its location and select it.

File of type

Specify the file type displayed in the dialog box.

To access this dialog: from the **Project** menu, select **Open Project**.

Organize Constraints for Synthesis Dialog Box

The Organize Constraints for Synthesis dialog box enables you to specify the order of your constraint files in your synthesis tool. Synthesis supports only SDC files.

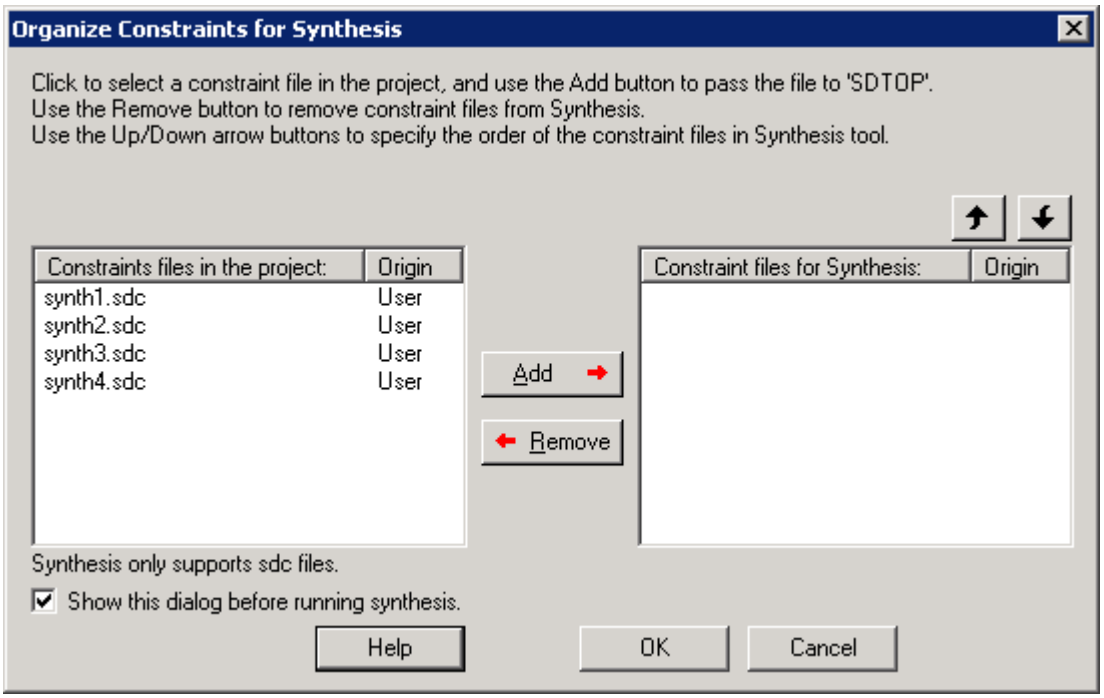


Figure 40 · Organize Constraints for Synthesis Dialog Box

Constraints files in the project / Origin

Lists all the synthesis constraint files in your current Libero IDE project and the origin of the file.

Constraint files for Synthesis / Origin

Lists the constraint files you have added for synthesis and their origin. The synthesis order proceeds from the top of the list to the bottom. Use the Up and Down arrows to change the order of the files.

Add

Adds files to the Constraint files for Synthesis list in your project.

Remove

Removes files from the Constraint files for Synthesis list in your project.

Up and **Down Arrows**

Use these buttons to order the stimulus files.

De-select the **Show this dialog before running synthesis** checkbox if you do not wish to organize your constraints each time you run synthesis.

Organize Stimulus Dialog Box

Use the Organize Stimulus dialog box (below) to manage the stimulus files in your project.

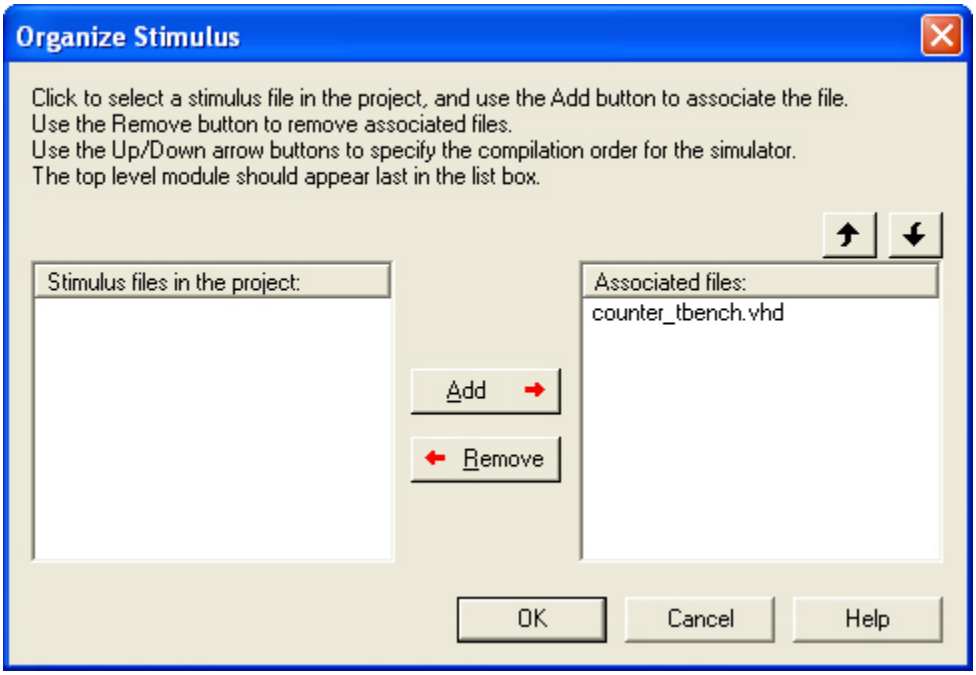


Figure 41 · Organize Stimulus Files Dialog Box

Stimulus files in the project

Lists all the stimulus files in your current Libero IDE project. In most cases you will only have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the *Associated Files* list box.

Associated files

Lists all the files associated with your project.

Add

Adds files to the Associated files list in your project.

Remove

Removes files from the Associated files list in your project.

Up and **Down Arrows**

Use these buttons to order the stimulus files. The top-level entity should be at the bottom of the list.

Page Setup Dialog Box

Use this dialog box to format your page header and footer. The default displays your project file name in the header and the page number of the total number pages (such as "1 of 4") in the footer.

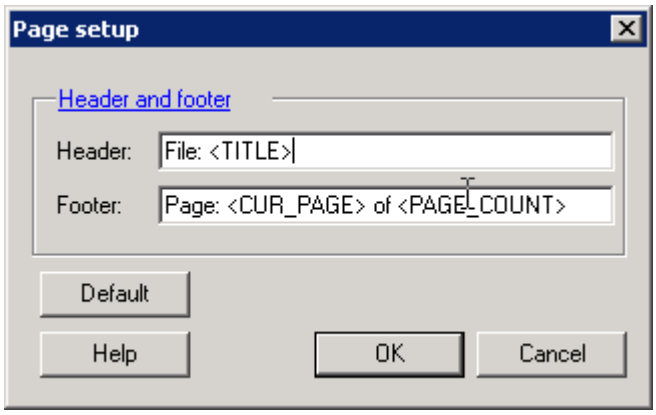


Figure 42 · Page Setup Dialog Box

You can add additional information about your project, such as a project description, the names of team members, or the date you started. To do so, type the information into the Header (or Footer) field.

The Default button resets the fields to their original settings.

Project Profile Dialog Box

Each Libero IDE project can have a unique and different profile, enabling you to integrate different tools for each Libero project. You can also export your project profile from Libero IDE to use it with a different project.

For example, if you get an updated version of your simulation tool you can change your project profile in one project, then export your profile. Importing the updated profile into a different project enables you to use your new simulation tool immediately.

A red question mark next to a profile means that the location or version of the tool has changed. You must click Edit to specify the location of the tool.

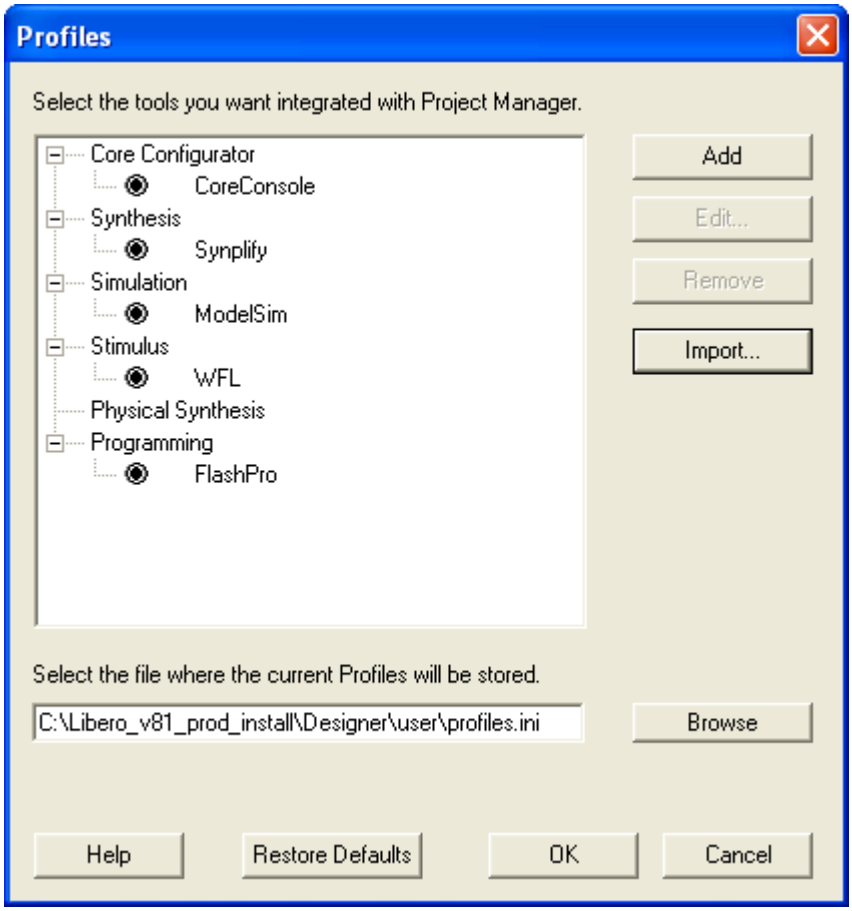


Figure 43 · Project Profile Dialog Box

Add

Click **Add** and select which type of tool (synthesis, stimulus, or simulation). Fill out the tool profile and click **OK**.

Edit

After selecting the tool, click to **Edit** to select another tool, change the tool name, or change the tool location.

Remove

After selecting a tool, click **Remove.**

Import

Imports an existing tool profile; navigate to the INI file and click OK to add it to your project.

Select the file where the current Profiles will be stored

Specifies the location of your profiles.ini file; the profiles.ini file stores all your project profile information. If you wish to change the location, enter a pathname or click the Browse button to specify a new location.

Restore Defaults

Restores your default project profile settings.

To access this dialog, from the **Project** menu, choose **Profile**.

See Also

Synplify AE

Activating PALACE for physical synthesis

WaveFormer Lite

ModelSim AE

Save Project As Dialog Box

The Save Project As dialog box enables you to save your entire project (Designer views, tool profiles, etc.) with a new name and location. save

Enter the name and location for your modified project and click OK to continue.

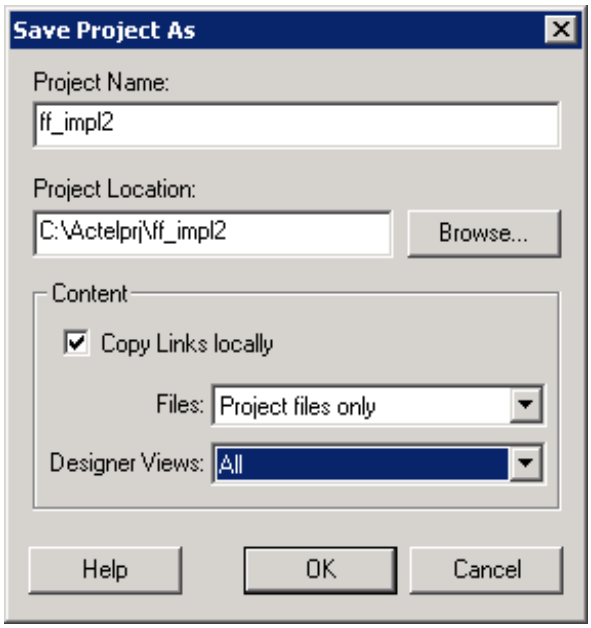


Figure 44 · Save Project As Dialog Box

Project Name

Type the project name for your modified project.

Project Location

Accept the default location or **Browse** to the new location where you can save and store your project. All files for your project are saved in this directory.

Content

Copy Links locally - Select this checkbox to copy the links from your current project into your new project. If you do not select this checkbox, the links will not be copied and you must add them manually.

Files - Select All, Project files only, Source files only, or None (empty project) to save your project with only the files you wish to keep.

Designer Views - Select All, Current view, or None to copy your views into your new project.

To access this dialog, from the **Project** menu, choose **Save Project As.**

Script Export Options Dialog Box

If you export a Tcl script in the Project Manager, the Script Export Options dialog box appears.

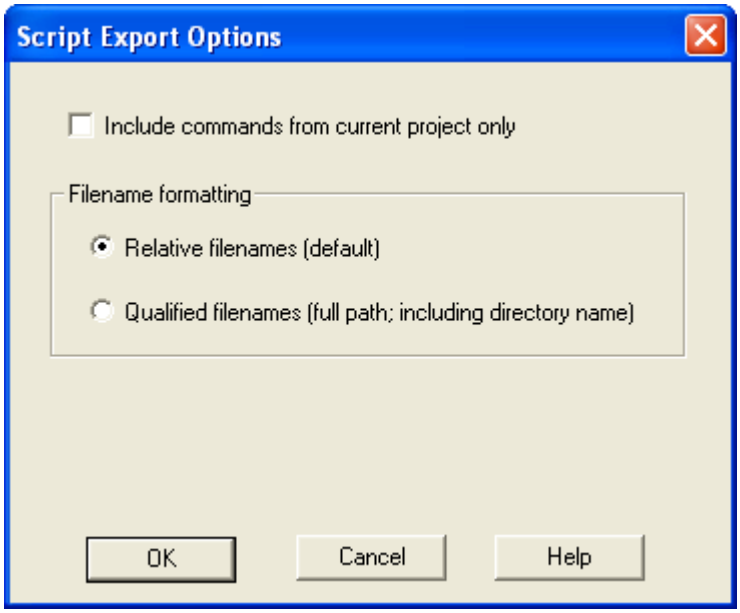


Figure 45 · Script Export Options Dialog Box

Include commands from current project only - Select this option if you want to include all the commands from your current project.

Filename Formatting - Choose **Relative filenames** if you do not intend to move the Tcl script from the saved location, or **Qualified filenames** if you plan to move the Tcl script on your machine.

Project Manager Project Menu

Command	Sub-menu	Icon	Function
New Project			Starts the New Project Wizard
Open Project			Opens the Open Project dialog box
Close Project			Closes the current active project; the Project Manager remains open
Save Project			Saves the current project
Save Project As			Saves the current project in a new directory; prompts you to enter a new project name
Detect New Files			Detects new files and adds them to the Project Manager (if found).
VHDL Library >	Add		Add a VHDL library in your project
	Remove		Remove a VHDL library in your project
	Rename		Rename a VHDL library in your project
Settings			Opens the Project Settings dialog box; enables you to manage device, simulation, and programming options for your project
Profile			Opens the Project Profile dialog box; enables you to specify locations for your third-party synthesis, stimulus, and simulation tools. Libero IDE includes tools for synthesis, physical synthesis, stimulus, and simulation.
Configure Project Flow			Opens the Configure Flow dialog box, sets options on your HDL netlists, DRC checker, ViewDraw, and a resource report when you generate a core.
File Organization	Source Files for Synthesis		Opens the Organize Source Files for Synthesis dialog box

Command	Sub-menu	Icon	Function
	Source Files for Simulation		Opens the Organize Source Files for Simulation dialog box
	Designer Constraint Files		Opens the Organize Constraints for Designer dialog box; enables you to organize your Designer constraint files in the Project Manager
	Designer CDB Files		Opens the CDB File Organization dialog box; the CDB File Organization dialog box manages conflict resolution (if any) between the blocks in your design
	Stimulus		Opens the Organize Stimulus dialog box; enables you to organize your stimulus files in the Project Manager
Designer Views >	Select		Choose your Designer View from the drop down menu. Note that views are listed in they order they were created
	Previous		Switches to the previous view
	Next		Switches to the next view
	Add		Opens the Add Designer View dialog box; enables you to specify the name of your latest view
	Remove		Opens Remove Designer View dialog box; enables you to delete all related Designer View files from your disk
	Rename		Opens Rename Designer View dialog box; select the Designer View you want to rename and enter the new name
Execute Script			Opens Execute Script dialog box; enables you to run Tcl script from the Project Manager
Preferences			Opens the Preferences dialog box
Recent Projects			Opens list of recent Libero IDE projects.
Exit			Closes Project Manager

Project Manager File Menu

Command	Icon	Shortcut	Sub-menu	Function
New		Ctrl + N		Opens the Project Manager New file dialog box and prompts you to enter a name and specify a file type
Open		Ctrl + O		Opens the Project Manager Open dialog box; enables you to select a file to open
Close				Closes the current file; the Project Manager remains open
Save		Ctrl + S		Saves the current file
Save As				Saves the current file as a different type (such as a TXT file)
Import Files				Opens the Import Files dialog box; enables you to import project files into the Project Manager
Create Link				Opens the Create Link dialog box; browse to select the file you wish to link. Linked files are added to the Design Explorer in the Modules defined in multiple files list.
Change All Links				Opens the Change All Links dialog box; enables you to update/change all the links for the files in your project at once.
Unlink All: Copy Files Locally				Copies all linked files to your local project.
Export >			Profiles	Opens the Project Manager Export dialog box, and defaults to save the exported file with the profile (INI) extension.
			Script Files	Opens the Project Manager Export dialog box, and defaults to save the exported file with the Tcl (TCL) extension.
			Template	Opens the Project Manager Export dialog box, and defaults to save the exported file as a template with a VHD or VLG extension.
Publish Designer				Opens the Publish Designer Block dialog box

Command	Icon	Shortcut	Sub-menu	Function
Block				
Print				Displays the Print dialog box (if available); allows you to print whatever element of the project you are working on
Print Preview				Previews your selection before you print (if available)

Project Manager Edit Menu

Command	Sub-menu	Icon	Shortcut	Function
Undo			CTRL + Z	Reverses your last action
Redo			CTRL + Y	Reverses the action of your last Undo command
Cut			CTRL + X	Removes the selection from your design
Copy			CTRL + C	Copies the selection to the Clipboard
Paste			CTRL + V	Pastes the selection from the Clipboard
Select All			CTRL + A	Selects all the content in your current file (such as in a VHDL file open in the VHDL editor)
Find and Replace >	Find		CTRL + F	Displays the Find dialog box, which you use to locate instances, nets, ports, and regions
	Find Next		F3	Finds the next occurrence of the text in the Find field
	Replace		CTRL + H	Displays the Replace dialog box; enables you to search and replace content in your files (files must be open and selected to use this feature)
	Find in Files			Opens the Find in Files dialog box; enables you to search for text in files or directories that you specify
	Find Module / Component			Opens the Find Module dialog box; searches the Libero IDE project for specific entity (module) names

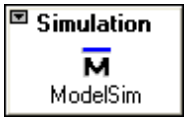
Command	Sub-menu	Icon	Shortcut	Function
Comment Out			CTRL + M	Inserts comment characters in the selected line (or lines) of your text editor
Uncommen t Out			CTRL + K	Removes comment characters from the selected line (or lines) of your text editor
Filter			CTRL + R	Refreshes the display

Project Manager View Menu

Command	Sub-menu	Shortcut	Function
Toolbars >	Status Bar		Shows/hides status bar in Project Manager
	Standard		Shows/hides standard toolbar in Project Manager
	Designer Views		Shows/hides Designer views toolbar in Project Manager
	Canvas		Available only when using SmartDesign; shows/hides the Canvas toolbar in Project Manager
	Top Symbol		Available only when using SmartDesign; shows/hides the top symbol toolbar in Project Manager
	Rotate		Available only when using SmartDesign; shows/hides the Rotate toolbar in Project Manager
	Align		Available only when using SmartDesign; shows/hides the Align toolbar in Project Manager
	Shape		Available only when using SmartDesign; shows/hides the Shape toolbar in Project Manager
	Zoom		Available only when using SmartDesign; shows/hides Zoom toolbar in Project Manager
	Schematic		Available only when using SmartDesign; shows/hides Schematic toolbar in Project Manager
	SmartDesign Tools		Available only when using SmartDesign; shows/hides SmartDesign Tools toolbar in Project Manager

Command	Sub-menu	Shortcut	Function
	Design Explorer		Shows/hides Design Explorer window in Project Manager
	Catalog		Shows/hides Catalog window in Project Manager
	Information Window		Shows/hides Information Window in Project Manager
	Find Window		Shows/hides Find window in Project Manager
	Log Window		Shows/hides Log window in Project Manager
	Configure Information Window		Enables you to show/hide elements in the Information Window, such as lists of new features, or properties of your current project.
Project Flow Window			Shows/hides the Project Flow window in the Project Manager
Bring Project Flow Window to the front		F2	Moves Project Flow window to the front of display; useful if you have several windows open simultaneously.
Refresh Design Hierarchy		CTRL + R	Updates the Hierarchy tab in the Design Explorer window. Useful if you add files to the project and Project Manager does not show them in the Hierarchy.
Reset Window Layout			Returns Project Manager window layout to default.
Maximize Work Area		CTRL+W	Hides the Catalog, Log Window, and Design Explorer windows (if open) and expands the selected tab in the Project Flow or SmartDesign work area.
Zoom >	Zoom In	CTRL + +	Zooms into the view in the Project Manager (if enabled)
	Zoom Out	CTRL + -	Zooms out of the view in the Project Manager (if enabled)
	Zoom Region	CTRL +W	Zooms into a region of the view in the Project Manager (if enabled); click and drag to select a region.
	Zoom Fit	CTRL + 0	Fits the view into the Project Manager (if enabled)
	Pan		Scrolls the view in the Project Manager (if enabled)

Project Manager Tools Menu

Command	Sub-menu	Icon	Function
HDL Editor			Opens the New file dialog box and defaults to VHDL entity. Enter a filename and click OK to open the VHDL editor.
SmartDesign			Opens the New file dialog box and defaults to SmartDesign component. Enter a filename and click OK to open SmartDesign.
CoreConsole			Opens the New file dialog box and defaults to CoreConsole component. Enter a filename and click OK to open CoreConsole.
ViewDraw			Opens the New file dialog box and defaults to Schematic. Enter a filename and click OK to open ViewDraw.
ViewDraw > Tools	Generate EDIF		Creates a new EDIF file from the schematic
	Generate HDL		Creates a new HDL file from the schematic
Synplify Synthesis			Starts Synplicity Synplify
PALACE Physical Synthesis			Starts PALACE for physical synthesis; physical synthesis is not available for all devices.
WaveFormer Stimulus			Starts WaveFormer and creates a stimulus file named after your project
ModelSim Simulation			Starts ModelSim AE and opens any existing simulation files in your project

Command	Sub-menu	Icon	Function
Designer Place-and-Route			Starts Designer; enables you to compile, layout, and analyze your design.
Silicon Sculptor			Starts the Silicon Sculptor programming tool
FlashPro			Starts the FlashPro programming tool
Silicon Explorer			Starts the Silicon Explorer debug tool (if available)
Identify Debugger			Opens the Identify Debugger (from Synplify)
Publish Designer Block			Opens the Publish Designer Block dialog box.

Reference

SmartDesign Menu

Command	Icon	Function
Show Canvas View		Displays the Canvas
Show Grid View		Displays the Grid
Show Schematic View		Displays the Schematic Viewer
Show Memory Map / Data Sheet		Displays the datasheet for the design
Check Connectivity		Runs the Connectivity Checker
Generate Design		Generates your SmartDesign component
Auto Connect		Auto-connects instances
Add Port		Adds a port to the top of the SmartDesign component

Project Manager Window Menu

The Project Manager Window menu controls the Design Flow, HDL Editor, and SmartDesign windows that open in the Project Manager.

Command	Function
Close	Closes the active window (HDL Editor or Design Flow window)
Close All	Closes all the open windows
Cascade	Cascades all open windows in the Project Manager
Tile Vertically	Tiles all open windows in the Project Manager vertically
Tile Horizontally	Tiles all open windows in the Project Manager horizontally
Project Flow	Shows project flow window in Project Manager
<*.vhd filename>	Shows VHD file, if open, in Project Manager

Project Manager Help Menu

The Help menu enables you to access the Libero IDE online help, reference manuals, check for updates, and view your license and version information.

Command	Icon	Function
Help Topics		Opens the Libero Project Manager online help
Reference Manuals		Opens the list of manuals (PDF files) available for Libero Project Manager, including links to library guides, PDF versions of the online help, and information about third party (OEM) tool guides.
Actel Technical Support		Displays the Actel customer support web page in your default browser
Actel Web Site		Displays the main Actel page in your default browser
Check for Software Updates		Checks for updates to the Libero Project Manager software
License Details		Displays detailed license information for your version of Libero IDE Project Manager
About Libero Project Manager		Displays version and release numbers for Libero IDE Project Manager

Dialog Boxes

EDIF Import Options

When importing an EDIF netlist, you must specify your EDIF flavor. Your choices are

- GENERIC - For any non-Viewlogic and Mentor Graphics netlists
- VIEWLOGIC - For Viewlogic EDIF netlists
- MGC - For Mentor Graphics EDIF netlists

Select the flavor that you wish to use with your design and click **OK**.

Select the **Do not show this dialog box during import** checkbox if you do not want to be asked about this option again.

Execute Script Dialog Box

You can use the Execute Script dialog box to run Tcl scripts from within Designer. You do not need to have a design open in order to run a script.

Specify a script file, enter Arguments (if necessary), and click Run to execute.

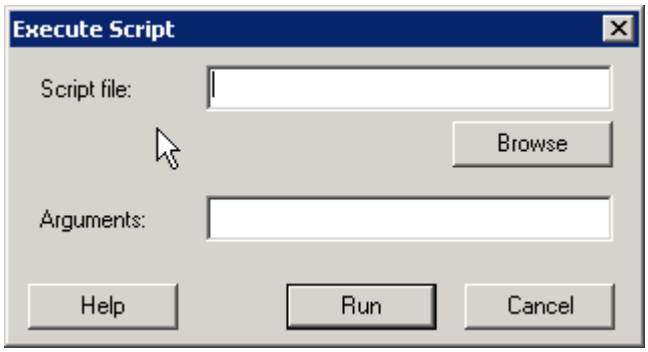


Figure 46 · Execute Script Dialog Box

Script file

Specify a script file. Browse to Select a script file with a valid extension (*.tcl or *.dsf).

Arguments

Input your arguments for your script file (if necessary).

Export Script / Log Files Dialog Box

The Export Script Files and Export Log Files dialog box both open from the File > Export menu in Designer. Select **File > Export > Script Files** or **Export > Log Files** to open the dialog box. The only difference is the files that display in the dialog boxes: Export Script Files displays only *.tcl files, and Export Log Files displays only *.log files.

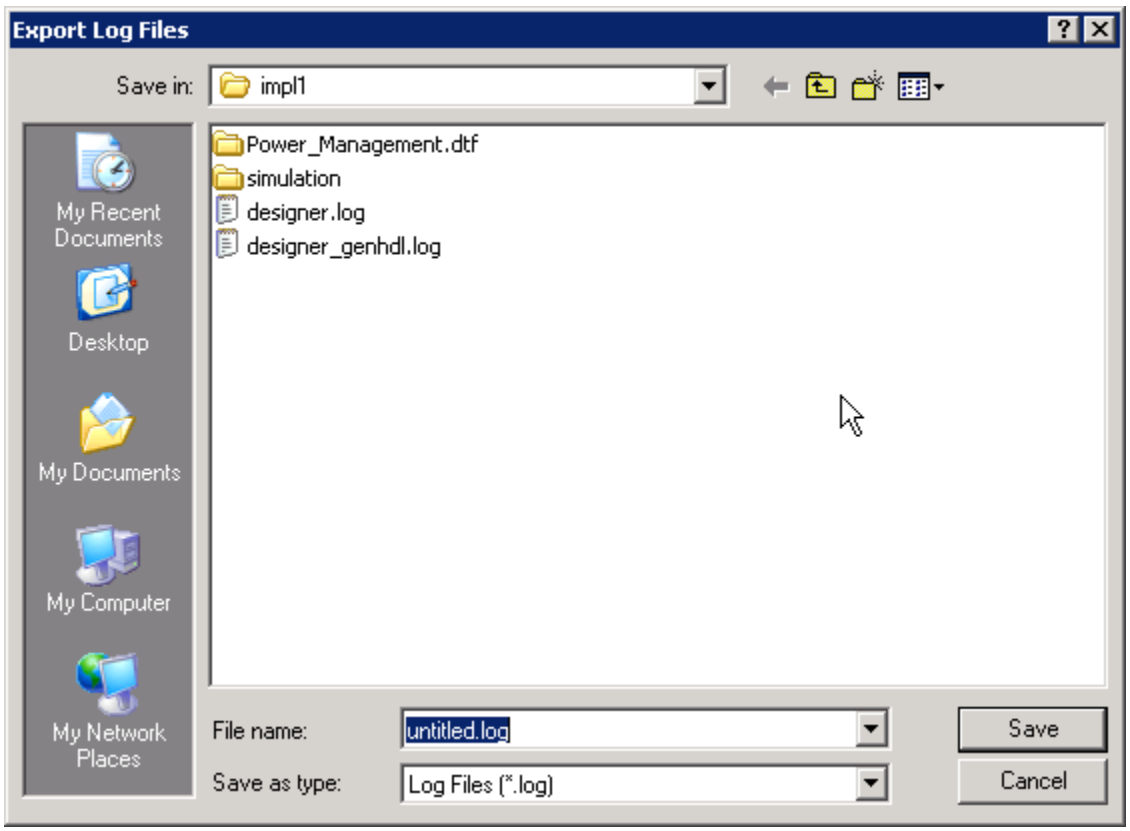


Figure 47 · Export Script / Log Files Dialog Box

Save in

Specifies the location of the file you are about to save.

File name

Specifies the filename of your new Tcl script or log file.

Save as type

Depending on your dialog box, saves the file as a LOG (*.log) file or a TCL (*.tcl) file

Generate a Programming File Dialog Box (Designer)

The Generate a programming file dialog box that appears depends on which device you are using.

If you are using a family that uses FlashPoint, the FlashPoint dialog box opens.

If you are using a family that uses a bitstream or STAPL file for programming, the dialog box is slightly different.

If you are generating a fuse file for programming an antifuse device (such as SX-A), the Generate a Fuse file dialog box appears.

Open Dialog Box (Designer)

Use the Open dialog box to open files in Designer. This dialog box appears when you select File > Open in Designer.

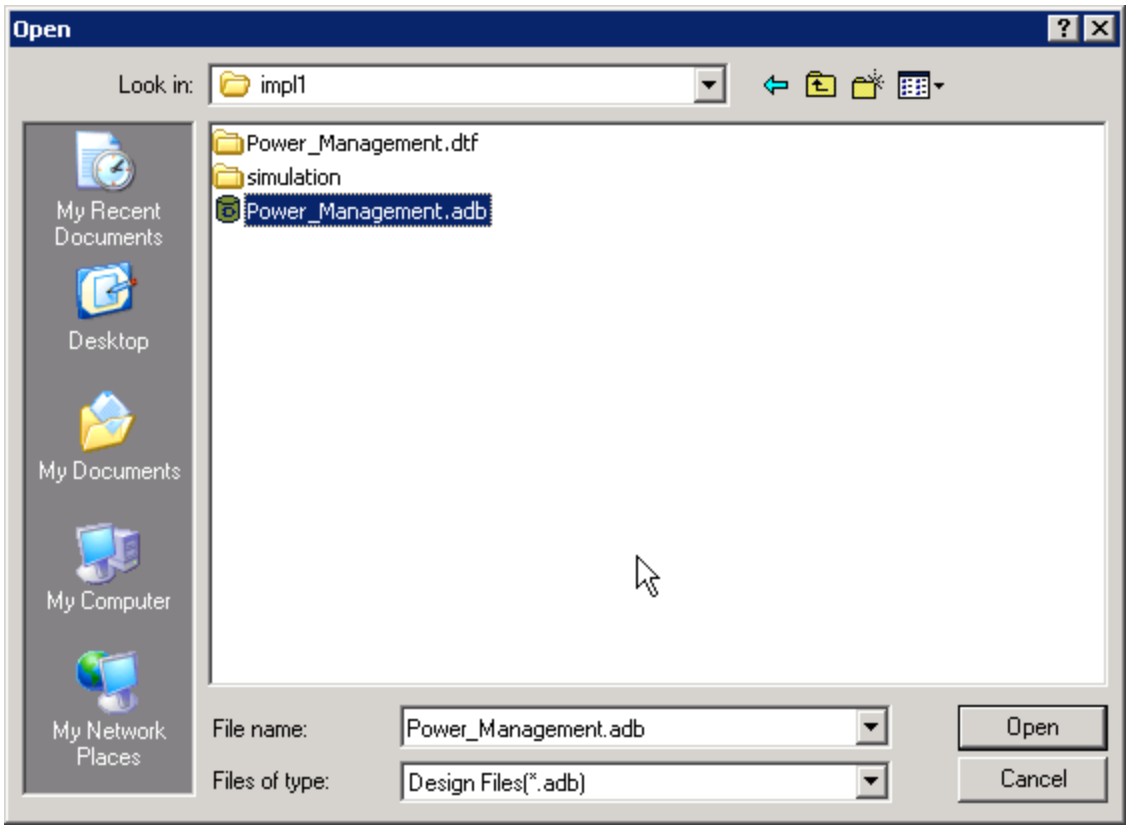


Figure 48 · Open Dialog Box (Designer)

Look in

Specifies from which directory you want to open the file. Navigate to a different directory if necessary.

File name

Specify a filename; type in part of a filename to filter files by that name.

Files of type

Select a file type from the drop-down menu to display only files of that type.

Setup Design Dialog Box (Designer)

The Setup Design dialog box opens when you select File > New, or when you select Tools > Setup Design. Use it to specify your design name, enable Designer Blocks, select your family, and specify your working directory.

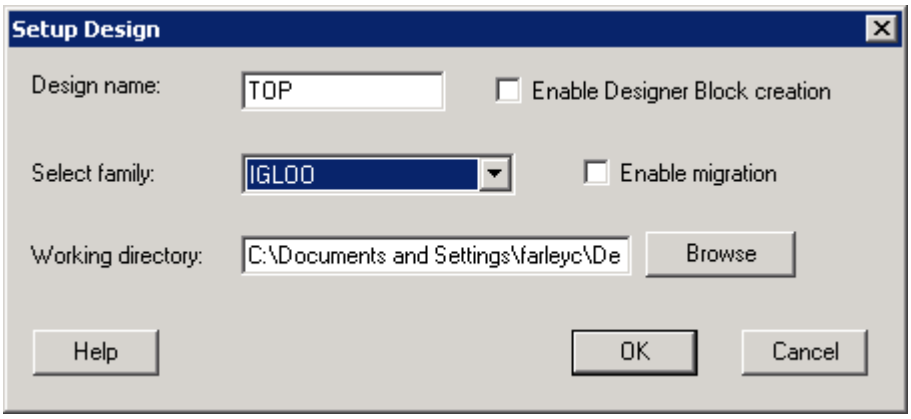


Figure 49 · Setup Design Dialog Box

Design Name

The name of the ADB file for your design. No spaces are allowed.

Enable Designer Block creation

Enables you to develop and publish Designer Blocks. Blocks are useful if you want to re-use the same elements in several designs. Once you place-and-route your block and publish, the performance for the block will not change. If you Enable Designer Block creation you cannot program your part, only publish your Designer Block.

Select Family

Choose or change your device family in the drop-down menu.

Enable migration (ProASIC3 only)

Click enable migration if you wish to migrate from a ProASIC3 device to an IGLOO device. The checkbox is enabled only if you are using a ProASIC3 device with a device and package that is compatible with IGLOO. If you choose to enable migration, the ProASIC3 to IGLOO Migration Options dialog box appears.

Working Directory

Set or change the working directory for your design.

Variables Console Dialog Box

The Variables Console provides direct access to internal settings of the Designer software. Although the Variables Console enables you to inspect and modify the internal settings, normally you do not need to do so, and making incorrect changes can result in adverse software behavior. Under certain circumstances, Actel Technical Support may walk you through instructions on how to view or change a setting using the Variables Console in Designer.

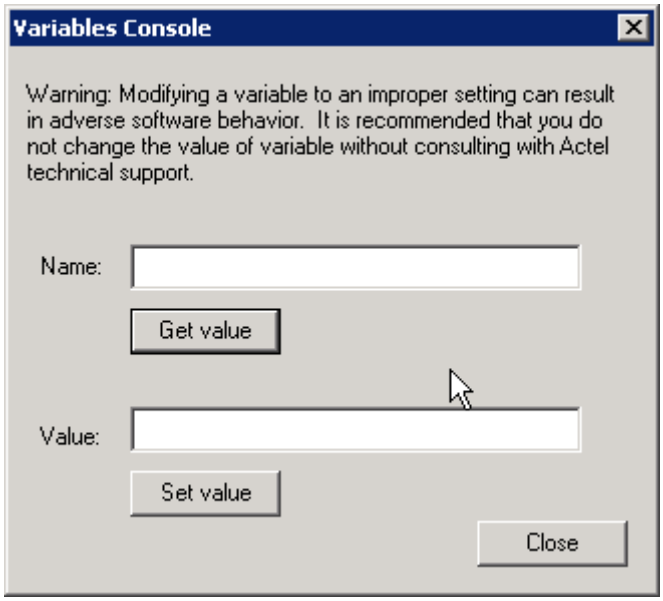


Figure 50 · Variables Console Dialog Box

Designer File Menu

Command	Icon	Shortcut	Function
New		CTRL + N	Opens the Setup Design dialog box and prompts you to enter a design name, select a family, and specify a working directory
Open		CTRL + O	Opens the Designer Open file dialog box; enables you to select a file to open
Close			Closes the current file
Save		CTRL + S	Saves changes to the current file
Save As			Enables you to save the file as a different type (such as TXT)
Execute		CTRL + U	Opens the Execute Script dialog box; use it to enter arguments and run

Command	Icon	Shortcut	Function
Script			Tcl scripts.
Import Source Files			Opens the Import Source Files dialog box; enables you to select and import source files (such as netlist and constraint files). Source files are audited to coordinate your design changes.
Import Auxiliary Files			Opens the Import Auxiliary Files dialog box. Aux files are not audited; if you make changes to an auxiliary file after you import it, Designer does not ask you to re-import the file.
Audit Settings			Opens the Audit Settings dialog box; enables you to view all the source files, change their location, and copy them to your local directory.
Export			Lists all the file types you can export in Designer.
Preferences			Opens the Designer Preferences dialog box
List of recent designs			Select a design from the list to open it.
Exit			Exits Designer

Designer View Menu

The View menu enables you to select which of the Designer elements you wish to display. You can choose to display (or hide) the standard tools (buttons for New, Open, Save, and About), the Design Tools (Compile, Layout, Back-annotate, Generate Programming File, and other tools), the Status Bar, and the Log Window.

Designer Tools Menu

The list of Tools varies according to your device family. Some tools, such as SmartTime and ChipPlanner, are not supported by all families. For a complete list of tools supported by each family, see the help for that tool.

Command	Icon	Function
Setup Design		Opens the Setup Design dialog box. Enter a design name, select a family, and specify a working directory for your design. You must save your design to commit your changes.
Device Selection		Opens the Device Selection Wizard so you can set your die, package, speed grade, die voltage, reserve pins, etc.
Compile		Compiles the design
Layout		Opens the Layout Options dialog box; select your options and click OK to run layout.
Back-Annotate		Opens the Back-Annotate dialog box; Back-Annotation creates the files necessary for back-annotation to the CAE file output type that you choose.
Generate Programming File		Opens the Generate Programming File dialog box; set your file type, silicon signature and output filename to generate the file.
Netlist Viewer		Starts Netlist Viewer, either on its own or in the MVN interface, depending on your device
PinEditor		Starts PinEditor, either on its own or in the MVN interface, depending on your device
ChipPlanner		Starts ChipPlanner in the MVN interface; use ChipPlanner to create, edit, and assign logic to regions in your design
ChipEditor		Starts ChipEditor
I/O Attribute Editor		Starts the I/O Attribute Editor in the MVN Interface; use this tool to view, sort, select, and delete I/O attributes in your design
SmartTime Constraints Editor		Starts the SmartTime Constraints Editor; use this tool to edit your timing constraints
SmartTime		Starts the SmartTime Timing Analyzer; use this tool to perform timing analysis on

Command	Icon	Function
Timing Analyzer		your design
SmartPower		Starts SmartPower; use this tool to analyze power usage in your design
Timer		Starts Timer; use this tool to perform timing analysis on your design
Reports		Opens the Report Types dialog box; select a report type and click OK to generate the report
Customize		Opens the Customize User Tools dialog box; use this dialog box to add other applications to the Designer Tools menu (PC only)

Designer Options Menu

Command	Function
Netlist Import	Enables you to select your Verilog and EDIF netlist options.
Compile	Opens the Compile options dialog box. This is not the same command as clicking the Compile button in Designer.
Variables Console	The Variables Console provides direct access to internal settings of the Designer software. Although the Variables Console enables you to inspect and modify the internal settings, normally you do not need to do so, and making incorrect changes can result in adverse software behavior. Under certain circumstances, Actel Technical Support may walk you through instructions on how to view or change a setting using the Variables Console in Designer.
Clear Log	Clears the Log window in Designer.

Designer Help Menu

The Help menu enables you to access the Designer online help, reference manuals, check for updates, and view your license and version information.

Command	Function
Help Topics	Opens the Libero IDE online help
Reference Manuals	Opens the list of manuals (PDF files) available for Designer, including links to library guides, and PDF versions of the online help.
Actel Technical Support	Displays the Actel customer support web page in your default browser
Actel Web Site	Displays the main Actel page in your default browser
Check for Software Updates	Checks for updates to the Designer software
License Details	Displays detailed license information for your version of Designer.
About Actel Designer Software	Displays version and release numbers for your Designer software

Device Programming

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.

IGLOO, Fusion, and ProASIC3 devices use the FlashPoint program file generator to create a programming file. The FlashPoint interface enables the advanced security features in all three device families.

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

Starting Silicon Sculptor from Libero IDE

Before starting Silicon Sculptor, generate your programming file.

To start the programming tool software:

1. Right-click the design root file in the **Hierarchy** tab.
2. Click **Run Silicon Sculptor.** Refer to the Silicon Sculptor User's Guide for information on using the programming tool.

Generating Programming Files

Generate a Programming File

FlashPoint enables you to program security settings, FPGA Array, and FlashROM features for IGLOO, Fusion, ProASIC3, and ProASIC family devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI.

You can generate a programming file with one, two, or all of the silicon features from the **Programming File Generator** first page.

To generate a programming file:

1. Select the **Silicon feature(s)** you want to program.
    - Security settings
    - FPGA Array
    - FlashROM

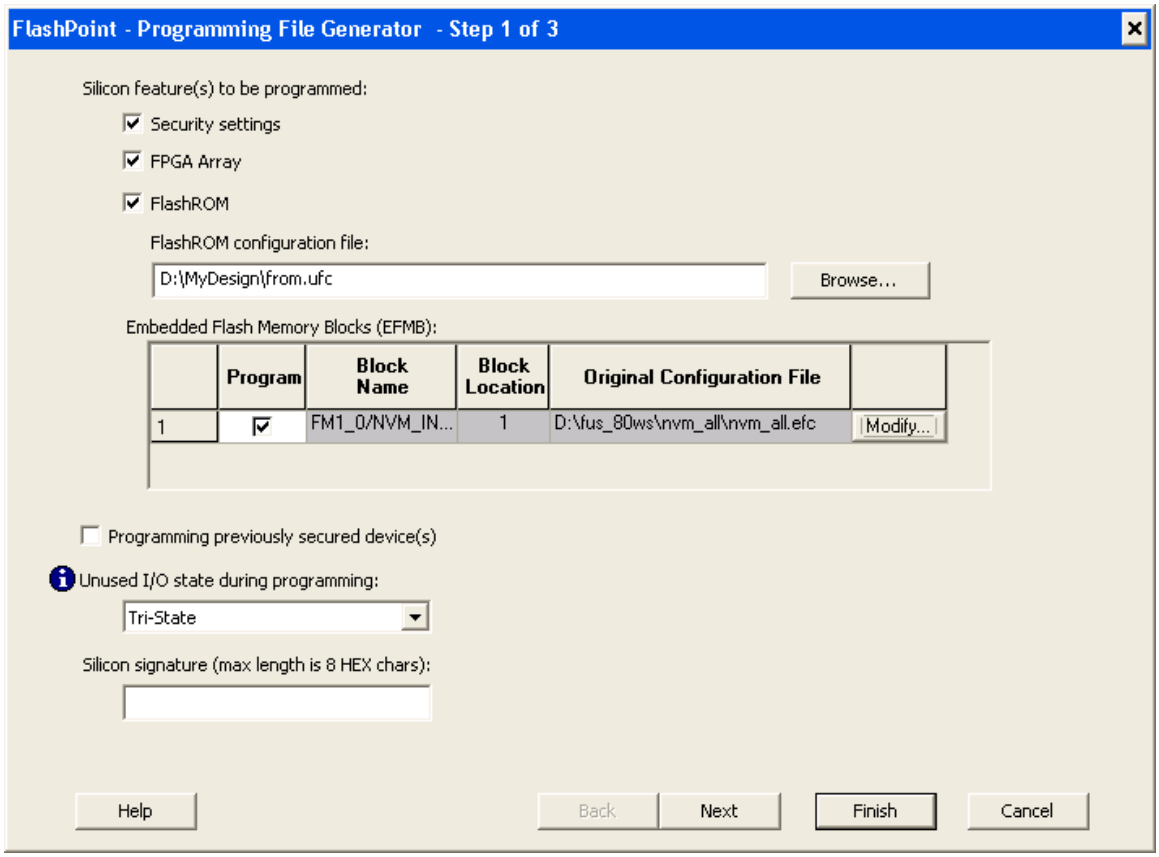


Figure 51 · Programming File Generator – Step 1 of 2

2. Click the **Programming previously secured device(s)** check box if you are reprogramming a device that has been secured.

Because the IGLOO, Fusion, ProASIC3, and ProASIC families enable you to program the Security Settings separately from the FPGA Array and/or FlashROM, you must indicate if the Security Settings were previously programmed into the target device. This requirement also applies when you generate programming files for reprogramming.

3. Specify the **Unused I/O state during programming** (Tri-State, Last Known State, High, or Low) in the drop down menu. See the table below for an explanation of these settings in the programming mode.

   Note: Unused I/O state during programming only applies to the unused I/Os; to modify the I/O state during programming for used I/Os, use the I/O Attribute Editor in the MultiView Navigator. For more information about changing the used I/O state during programming, see the MultiView Navigator online help.

Table 4 · I/O Settings in Programming Mode

Setting	Description
Tri-State	I/Os are tri-stated
Last Known State	I/Os are set to this state prior to entering the programming mode
High	I/Os are set to drive out logic High
Low	I/Os are set to drive out logic Low

4. Enter the silicon signature (0-8 HEX characters). See Silicon Signature for more information.
5. Depending upon the Silicon features you selected, click **Next** or **Finish**.

If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Save As** dialog box appears. From the **Save As** dialog box, choose STAPL file, SVF file, PDB file, 1532 file, or multiple files.

Generate a Programming File for CoreMP7/Cortex-M1 Device Support

FlashPoint enables you to program FPGA Array and FlashROM features for CoreMP7/Cortex-M1 devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI. You can generate a programming file with one, two, or all of the silicon features from the **Programming File Generator** first page. For CoreMP7/Cortex-M1 device support, you cannot select your own security settings. The generated programming file always has the encrypted FPGA Array content. The programming file generation is the same as the ProASIC3 family devices. ***To generate a programming file:***

1. Select the Silicon feature(s) you want to program.

FPGA Array

FlashROM

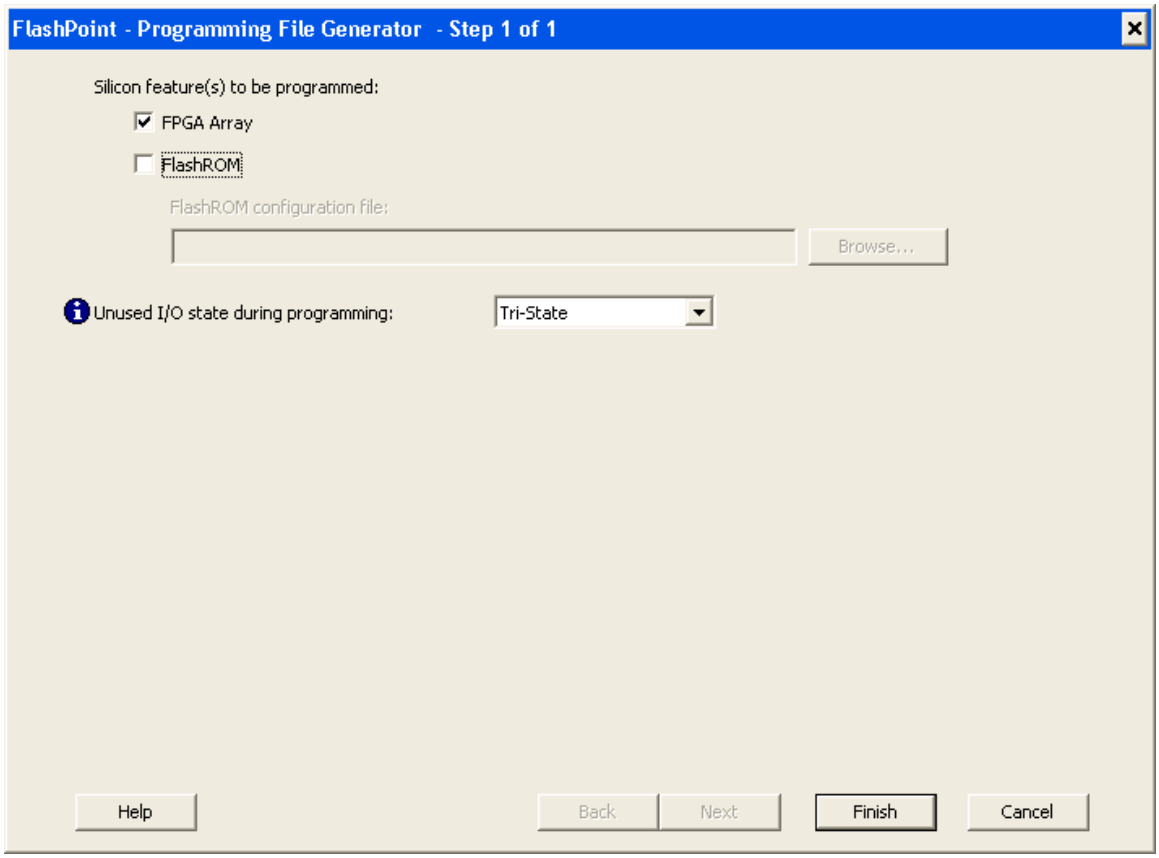


Figure 52 · FlashPoint- Programming File Generator for ProASIC3- Step 1 of 2

2. Specify the Unused I/O state during programming (Tri-State, Last Known State , High, or Low) in the drop down menu. See the table from Generate a programming file for an explanation of the settings in the programming mode.
3. Click **Next** or **Finished** depending on the silicon features you selected.

If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Save As** dialog box appears. From the **Save As** dialog box, choose STAPL file, SVF file, PDB file, 1532 file, or multiple files.

CoreMP7/Cortex-M1 Device Security

CoreMP7/Cortex-M1 devices are shipped with the following security enabled:

- FPGA Array enabled for AES encrypted programming and verification.
- FlashROM enabled for plain text read and write.

You cannot select your own security settings. The generated programming file includes the encrypted FPGA Array content.

Programming FlashROM and FPGA Array

For CoreMP7/Cortex-M1 device support, the programming generation for FlashROM and FPGA Array is the same as the programming generation for ProASIC3 and ProASIC family devices.

Generate a Programming File for AFS Device Support

FlashPoint enables you to program Security Settings, FPGA Array, Embedded Flash Memory Blocks, and FlashROM features for AFS device support. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI. You can generate a programming file with one, two, or all of the silicon features from the **Programming File Generator** first page.

AFS Programming

In addition to FPGA Array, FlashROM and security setting, the Fusion devices provide Embedded Flash Memory Blocks (FB) for both Analog configuration initialization and regular memory storage. Depending on the targeted AFS device, you may have one, two, or four FBs available to you. FlashPoint enables you to initialize the FB Instance(s) as specified in SmartGen. (Please refer to the SmartGen User's Guide for details).

To generate a programming file:

1. Select the **Silicon feature(s)** you want to program.

Security Settings

FPGA Array

FlashROM

Embedded Flash Memory Block

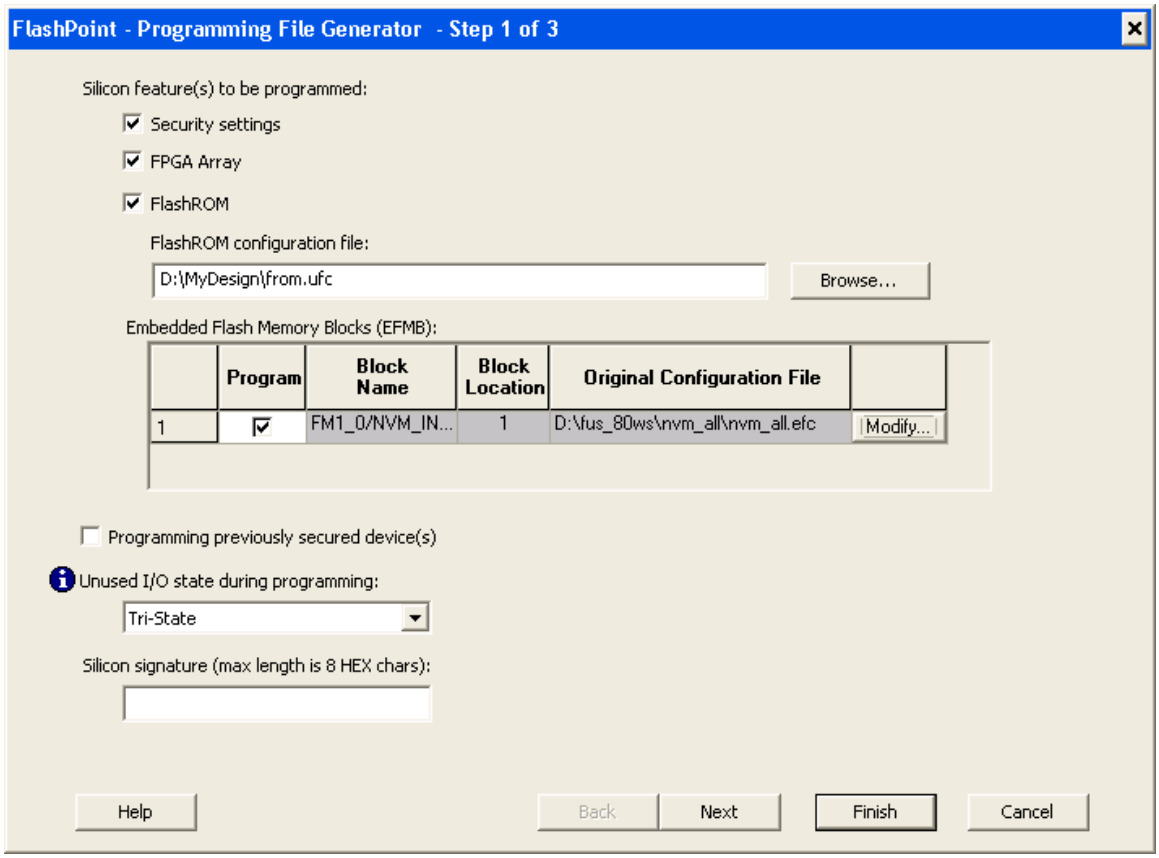


Figure 53 · FlashPoint- Programming File Generator for AFS

Note: Check the check box in the Program column to enable block modification.

2. Check the **Programming previously secured devices(s)** box if you want to program previously secured devices.
3. Specify the Unused I/O state during programming (Tri-State, Last Known State, High, or Low) in the drop down menu. See the table from Generate a programming file for an explanation of the settings in the programming mode.
4. Enter the **Silicon signature**.
5. Depending upon the Silicon features you selected, click **Finish** or **Next**.

If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Save As** dialog box appears. From the **Save As** dialog box, choose PDB file, STAPL file, SVF file, 1532 file, or multiple files.

Programming Security Settings, FlashROM, and FPGA Array

For AFS device support, the programming generation for Security Settings, FlashROM and FPGA Array is the same as the programming generation for ProASIC3 family devices.

Generate a Programming File for Serialization Support in In House Programming (IHP)

FlashPoint allows you to program security settings, FPGA Array, and FlashROM features for IGLOO, Fusion, ProASIC3, and ProASIC family devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI.

SVF Serialization Support in IHP

In addition to FPGA Array, FlashROM, and security setting, FlashPoint supports generating SVF files with serialization support in IHP.

To generate SVF with serialization support:

1. Select the **Silicon feature(s)** you want to program.
    - Security settings
    - FPGA Array
    - FlashROM
    - Programming Embedded Flash Memory Block
2. Import the UFC file which contains serialization data to FlashROM. See figure below. Click **Next**.

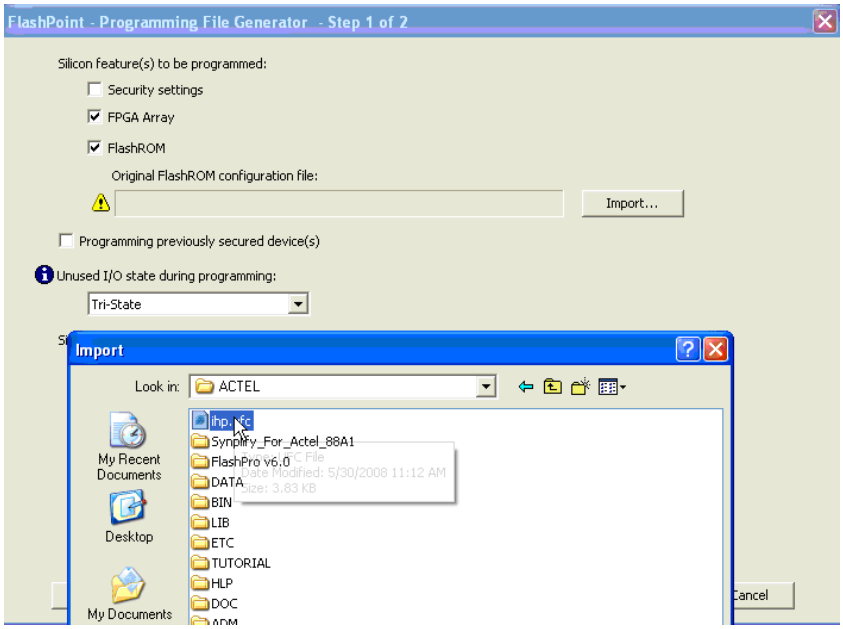


Figure 54 · Generate SVF File for Serialization Support in IHP

3. Type in the number of devices to program. See figure below.

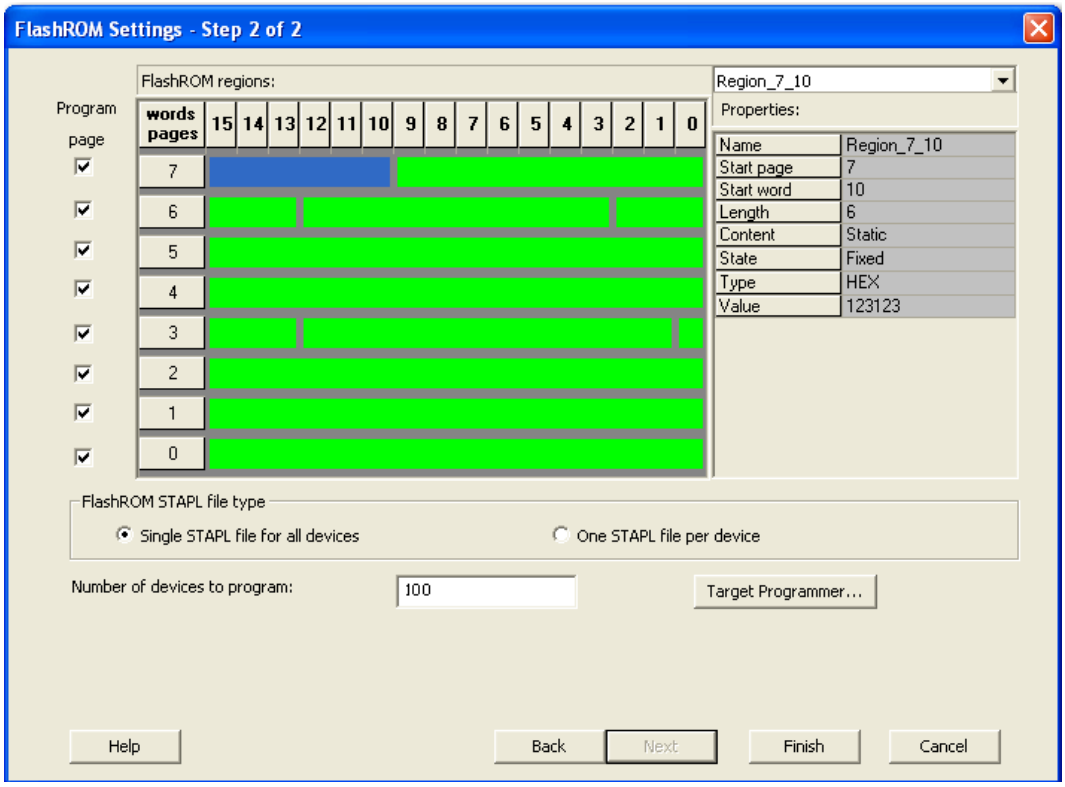


Figure 55 · Type Number of Devices

4. Click **Target Programmer** and select **Actel IHP**.

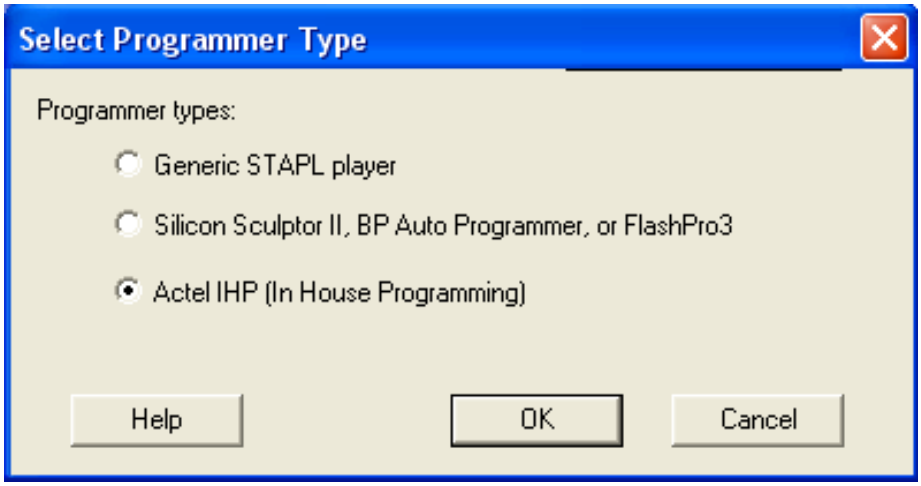
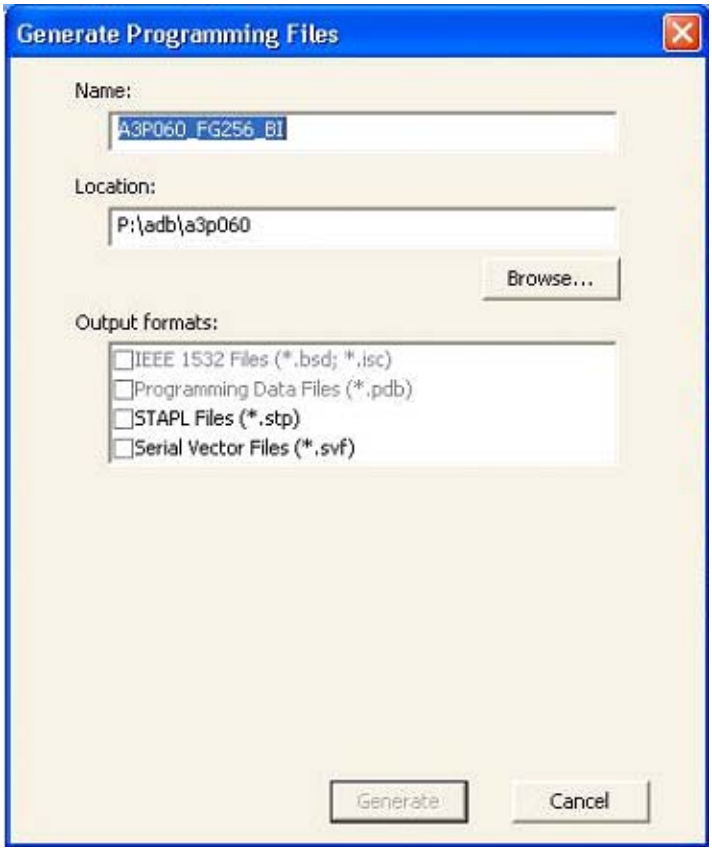


Figure 56 · Select Actel IHP

5. Click **OK**. A **Generate Programming Files** window displays. See figure below. Select **Serial Vector Files (*svf)**.



Select Serial Vector Files

6. Click **Generate**. An Actel specific SVF file will be generated with a corresponding serialization data file.

Note: Generated SVF files will only work with IHP.

Programming Embedded Flash Memory Block

For more information about the Embedded Flash Memory Block, see the Flash Memory System Builder online help.

To program the Embedded Flash Memory Block:

1. Check the **Program** box to enable Embedded Flash Memory Block modification.

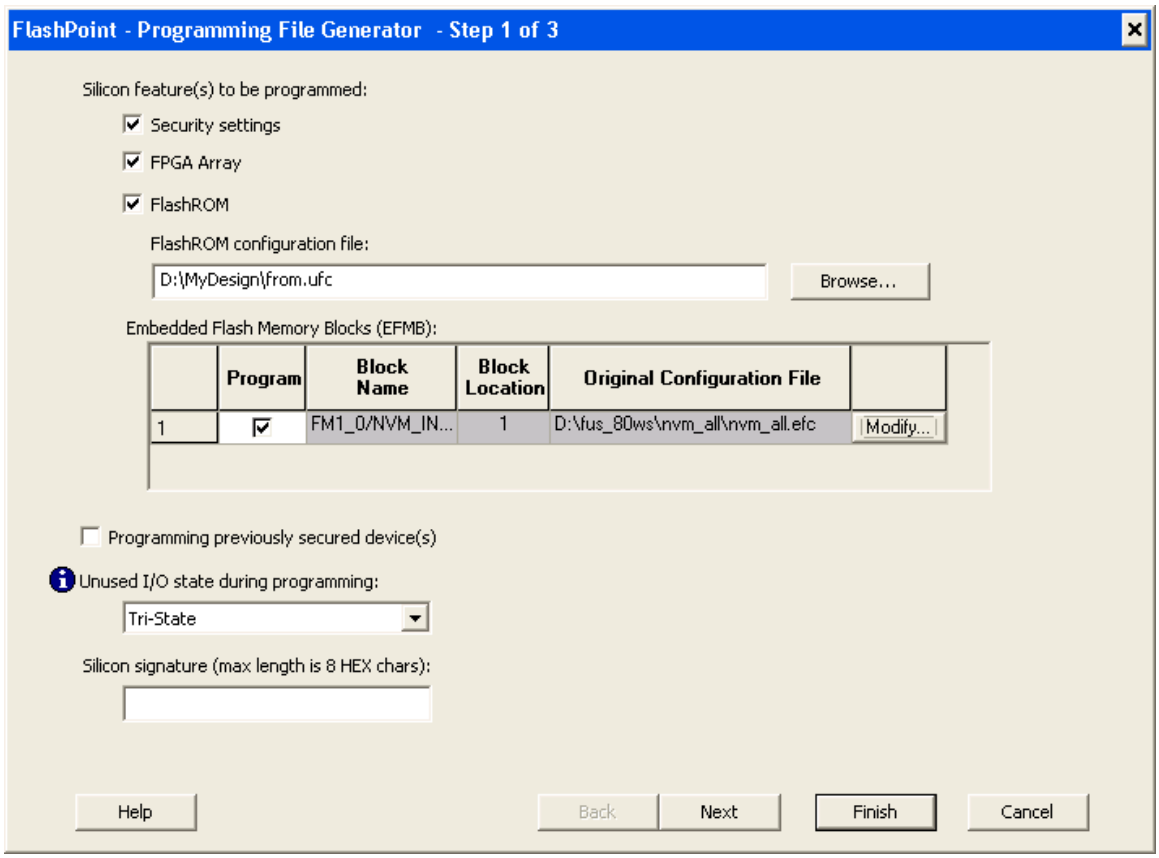


Figure 57 · Program File Generator first page

2. Click the **Modify** button to import Embedded Flash Memory Block configuration and memory content.

The **Modify Embedded Flash Memory Block** dialog box appears.

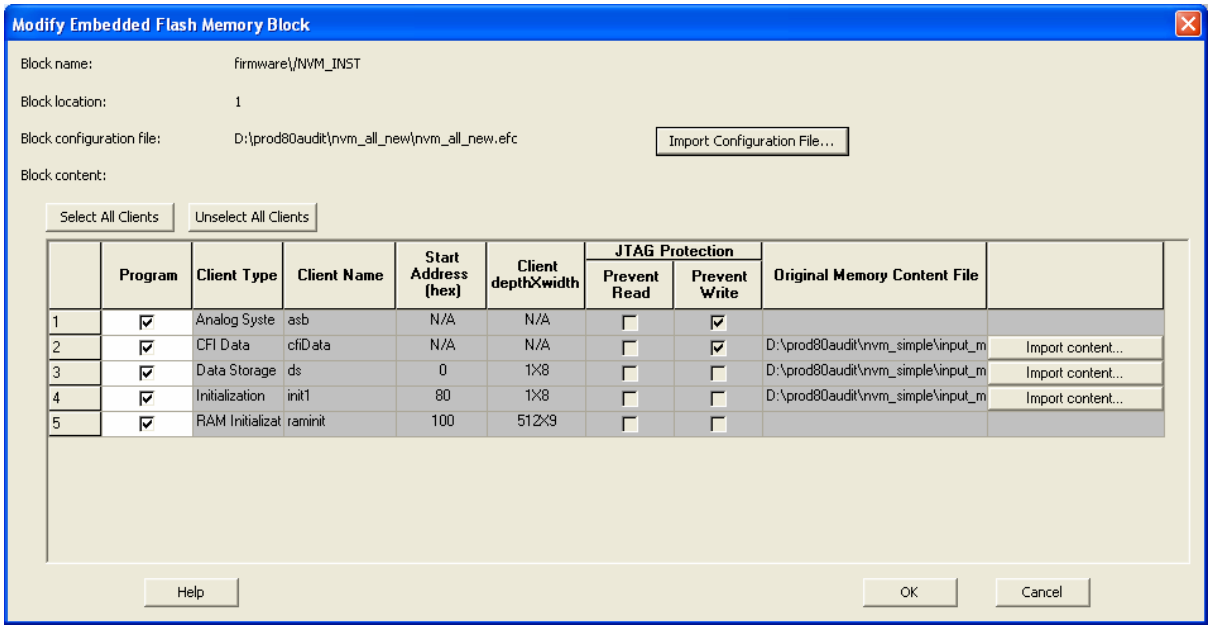


Figure 58 · Modify Embedded Flash Memory Block Content Dialog Box

3. Click the **Import Configuration File** button to import the Embedded Flash Memory Block configuration and memory content from the EFC file. This will populate the client table below. All clients that belong to this block will be selected by default.
4. Click the **Import content** button if you want to change the client memory content.
5. Click **OK**.

Note: FlashPoint audits original configuration and memory content files and warns the user if the files cannot be located or if they have been updated.

Programming the FPGA Array

You can program the FPGA Array by selecting the silicon feature **FPGA Array** in the Generate Programming File page and clicking **OK**.

See Generate a programming file for more information.

Programming the FlashROM

You can program selected memory pages and specify the region values of the FlashROM.

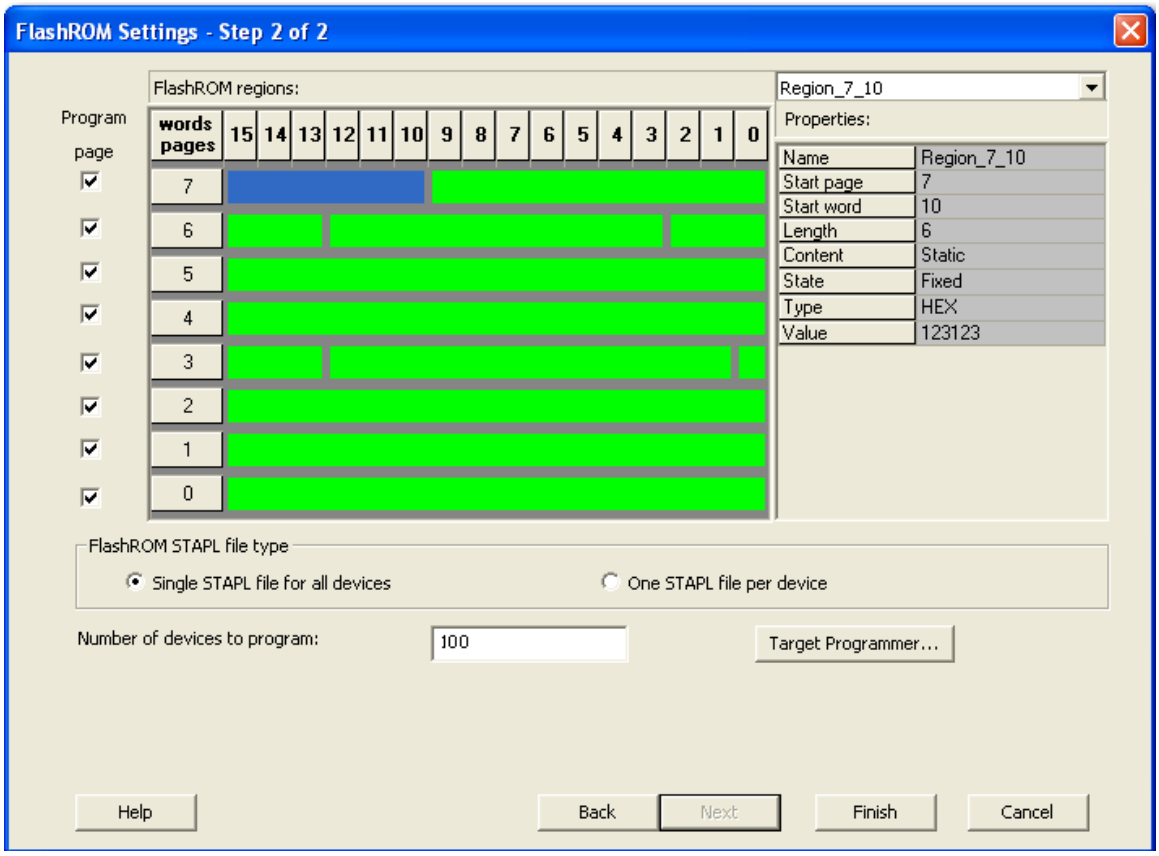


Figure 59 · FlashROM Settings

Single STAPL file for all devices: generates one programming file with all the generated increment values or with values in the custom serialization file.

One STAPL file per device: generates one programming file for each generated increment value or for each value in the custom serialization file.

1. Select your target **Programmer type**.

   • Select Generic STAPL Player when generating STAPL files for generic STAPL players.

   • Select Silicon Sculptor II, BP Auto Programmer, or FlashPro3 when generating programming files for those programmers.

   • Select Actel IHP (In House Programming) when generating STAPL or SVF files for Actel IHP.

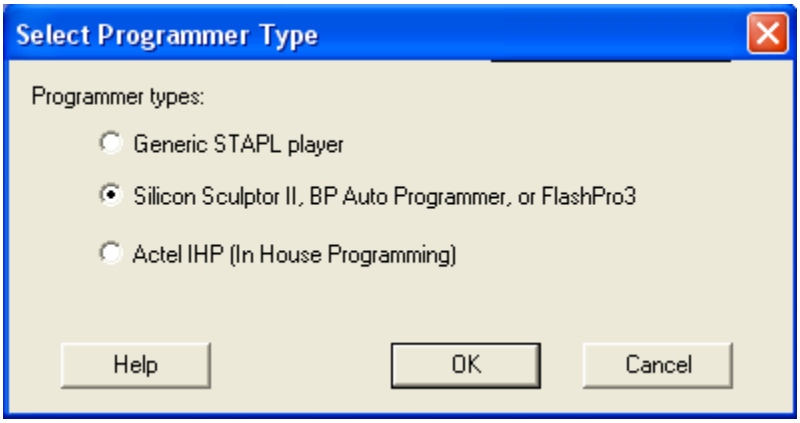


Figure 60 · Select Programmer

2. Click **OK**.

   FlashPoint generates your programming file.

Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the SmartGen FlashROM core generator.

For more information, see the SmartGen online help.

To program FlashROM:

1. Select **FlashROM** from the **Generate Programming File** page.
2. Enter the location of the FlashROM configuration file. The **FlashROM Settings** page appears (see figure below).

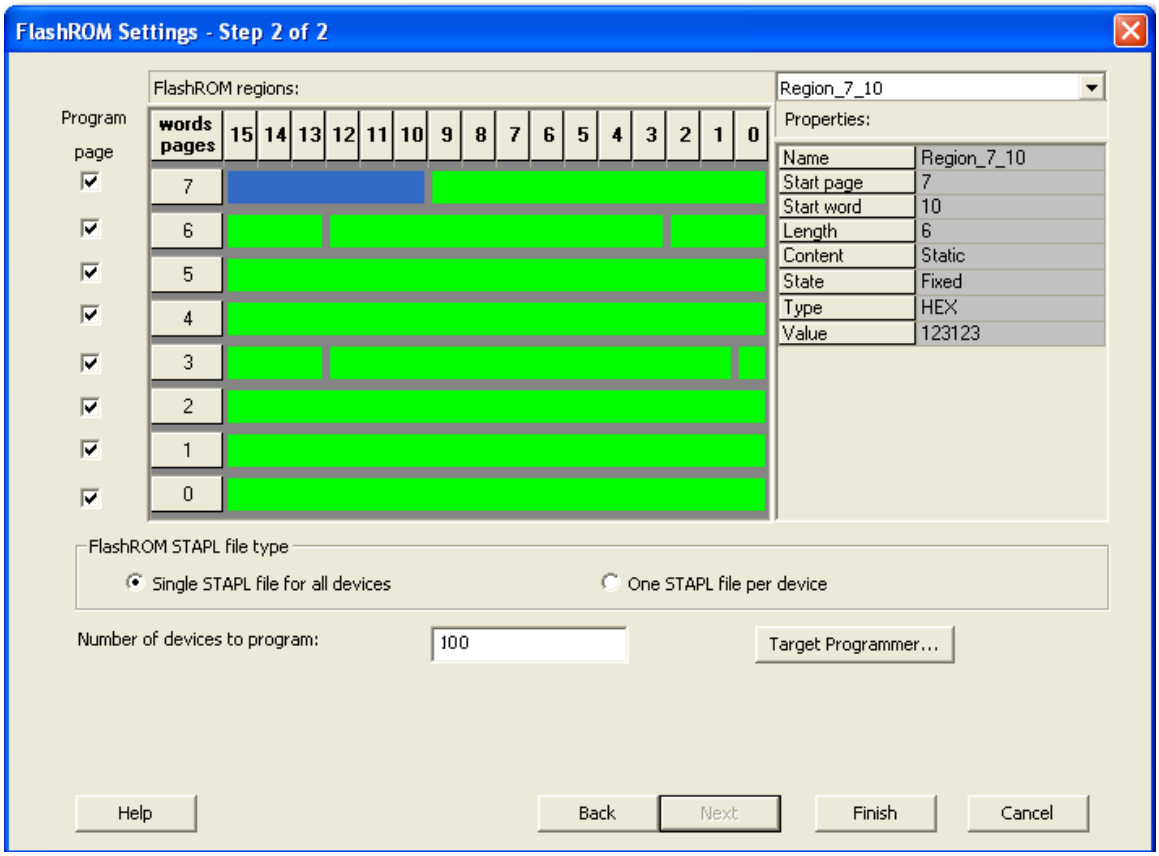


Figure 61 · FlashROM Settings

3. Select the FlashROM memory page that you want to program.
4. Enter the data value for the configured regions.
5. If you selected the region with a **Read From File**, specify the file location.
6. If you selected the **Auto Increment** region, specify the **Start** and **Max** values.

Enter the number of devices you want to program.

Select your target Programmer Type.

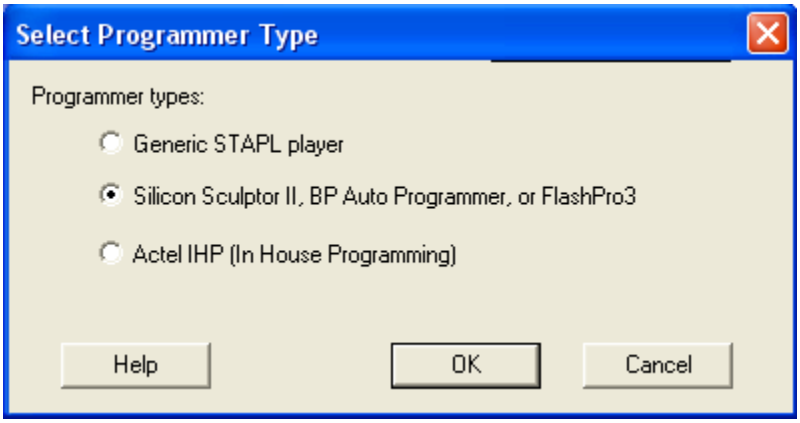


Figure 62 · Select Programmer

7. Click **Finish**.

   FlashPoint generates your programming file.

   Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the SmartGen FlashROM core generator.

For more information, see the SmartGen online help.

Silicon Signature

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

The silicon signature is accessible through the USERCODE JTAG instruction.

Note: If you set the security level to high, medium, or custom, you must program the silicon signature along with the Security Setting. If you have already programmed the Security Setting into the target device, you cannot reprogram the silicon signature without reprogramming the Security Setting.

Note: The previously programmed silicon signature will be erased if:

- You have already programmed the silicon signature and
- You are programming the security settings, but you do not have an entry in the silicon signature field

Programming Security Settings

FlashPoint allows you to set a security level of high, medium, or none.

To program Security Settings on the device:

1. If you choose to program Security Settings on the device from the **Generate Programming File** page, the wizard takes you to the **Security Settings** page (see figure below).
2. Move the sliding bar to select the security level for FPGA and FlashROM (see table for a description of the security levels).

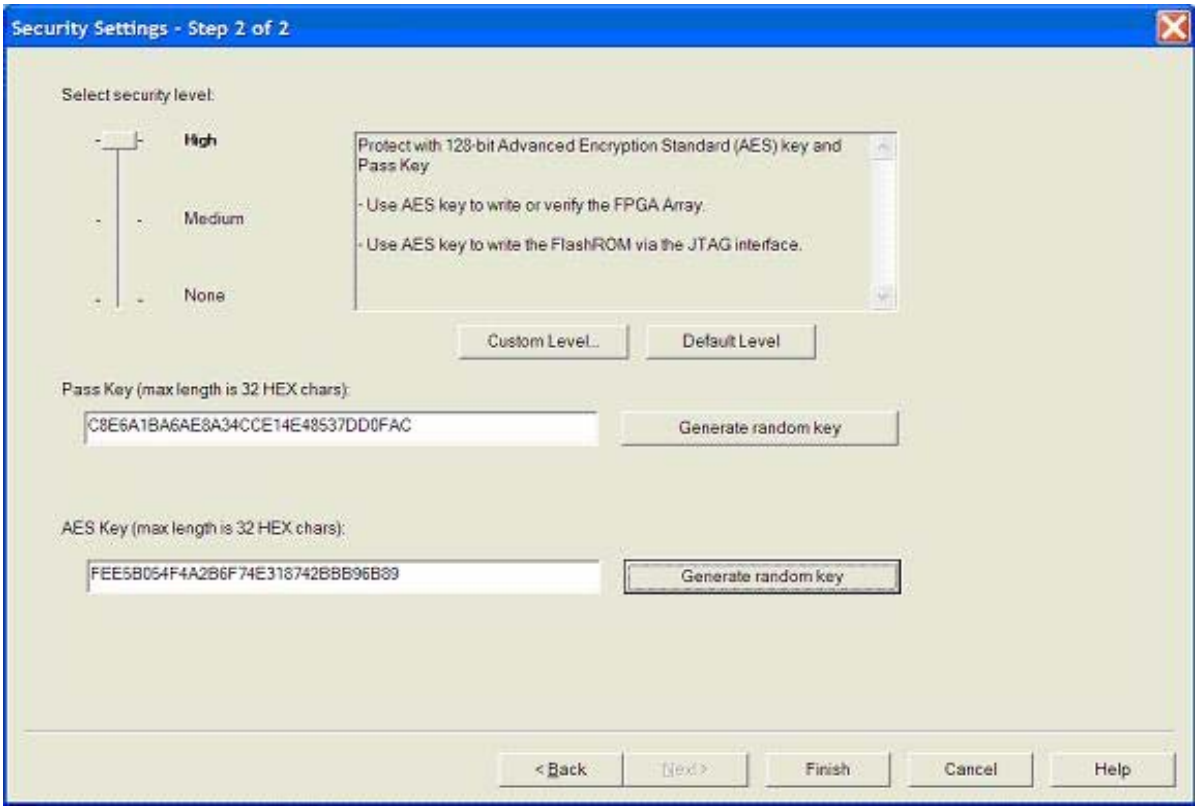


Figure 63 · Security Settings Page

Table 5 · FPGA and FlashROM Security Levels

Security Level	Security Option	Description
High	Protect with a 128-bit Advanced Encryption Standard (AES) key and a Pass Key	Access to the device is protected by an AES Key and the Pass Key. The Write and Verify operations of the FPGA Array use a 128-bit AES encrypted bitstream. From the JTAG interface, the Write and Verify operations of the FlashROM use a 128-bit AES encrypted bitstream. Read back of the FlashROM content via the JTAG interface is protected by the Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
Medium	Protect with Pass Key	The Write and Verify operations of the FPGA Array require a Pass Key.

Security Level	Security Option	Description
		From the JTAG interface, the Read and Write operations on the FlashROM content require a Pass Key. You can Verify the FlashROM content via the JTAG interface without a Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
None	No security	The Write and Verify operations of the FPGA Array do not require keys. The Read, Write, and Verify operations of the FlashROM content also do not require keys.

3. Enter the **Pass Key** and/ or the **AES Key** as appropriate. You can generate a random key by clicking the **Generate random key** button.

- The **Pass Key** protects all the Security Settings for the FPGA Array and/or FlashROM.

  The **AES Key** decrypts FPGA Array and/or FlashROM programming file content. Use the AES Key if you intend to program the device at an unsecured site or if you plan to update the design at a remote site in the future.

You can also customize the security levels by clicking the **Custom Level** button. For more information, see the Custom Security Levels section.

Custom Security Levels

For advanced use, you can customize your security levels.

To set custom security levels:

1. Click the **Custom Level** button in the **Setup Security** page. The **Custom Security** dialog box appears (see figure below).

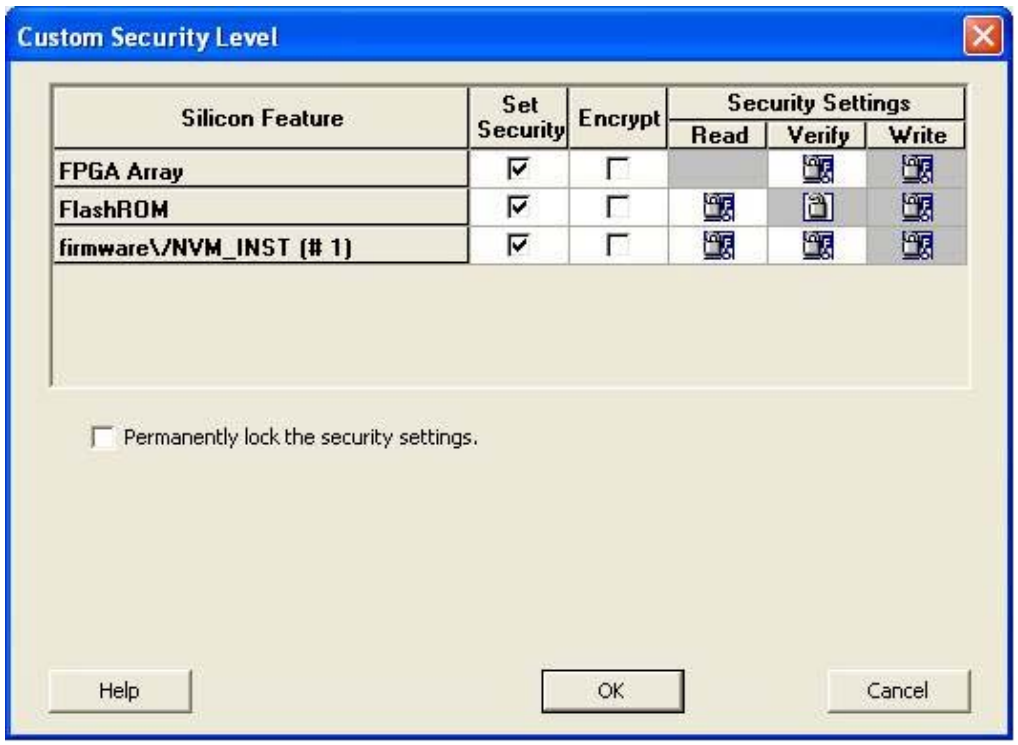
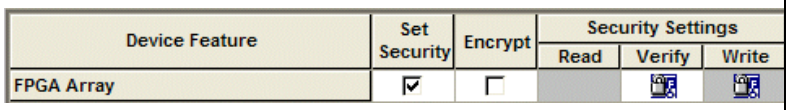
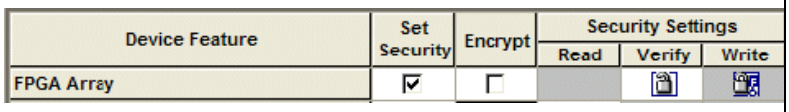
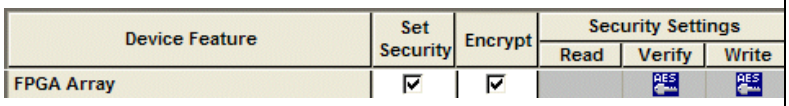
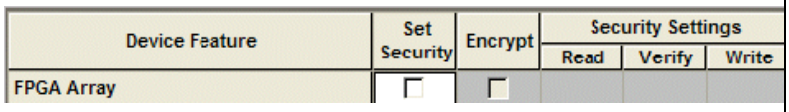


Figure 64 · Custom Security Level

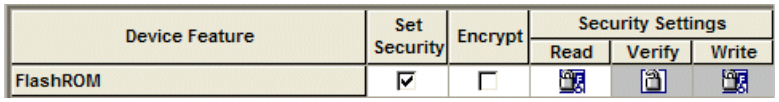
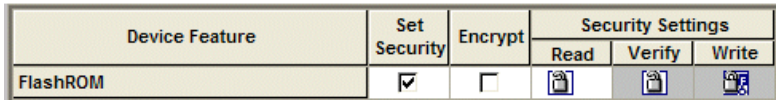
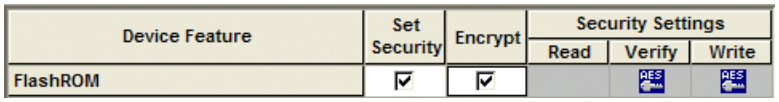
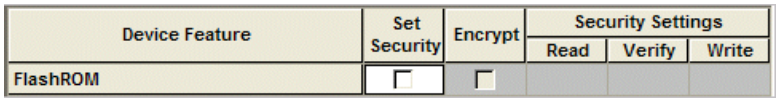
2. Select the **FPGA Array Security** and the **FlashROM Security** levels. For Fusion devices, you can also choose the Embedded Flash Memory Block level of security. The FPGA Array and the FlashROM can have different Security Settings. See the tables below for a description of the custom security option levels for FPGA Array and FlashROM.

Table 6 · FPGA Array

Security Option	Description
Lock for both writing and verifying  	Allows writing/erasing and verification of the FPGA Array via the JTAG interface only with a valid Pass Key.
Lock for writing  	Allows the writing/erasing of the FPGA Array only with a valid Pass Key. Verification is allowed without a valid Pass Key.
Use the AES Key for both writing and verifying  	Allows the writing/erasing and verification of the FPGA Array only with a valid AES Key via the JTAG interface. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FPGA Array. Use this option if you intend to complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. Accessing the device security settings requires a valid Pass Key.
Allow write and verify  	Allows writing/erasing and verification of the FPGA Array with plain text bitstream and without requiring a Pass Key or an AES Key. Use this option when you develop your product in-house.

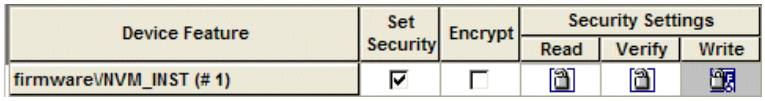
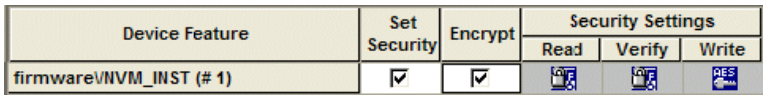
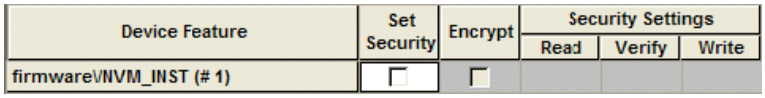
Note: The ProASIC3 family FPGA Array is always read protected regardless of the Pass Key or the AES Key protection.

Table 7 · FlashROM

Security Option	Description
Lock for both reading and writing 	Allows the writing/erasing and reading of the FlashROM via the JTAG interface only with a valid Pass Key. Verification is allowed without a valid Pass Key.
Lock for writing 	Allows the writing/erasing of the FlashROM via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.
Use the AES Key for both writing and verifying 	Allows the writing/erasing and verification of the FlashROM via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FlashROM. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. class="Note">Note: The bitstream that is read back from the FlashROM is always unencrypted (plain text).
Allow reading, writing, and verifying 	Allows writing/erasing, reading and verification of the FlashROM content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.

Note: The FPGA Array can always read the FlashROM content regardless of these Security Settings.

Table 8 · Embedded Flash Memory Block

Security Option	Description
Lock for reading, verifying, and writing 	Allows the writing and reading of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Verification accomplished by reading back and compare.
Lock for writing 	Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.
Use AES Key for writing 	Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogramming of the Embedded Flash Block. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. The bitstream that is read back from the Embedded Flash Memory Block is always unencrypted (plain text), when a valid pass key is provided.
Allow reading, writing, and verifying 	Allows writing, reading and verification of the Embedded Flash Memory Block content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.

3. To make the Security Settings permanent, select **Permanently lock the security settings** check box. This option prevents any future modifications of the Security Setting of the device. A Pass Key is not required if you use this option.

Note: When you make the Security Settings permanent, you can never reprogram the Silicon Signature. If you Lock the write operation for the FPGA Array or the FlashROM, you can never reprogram the FPGA Array or the FlashROM, respectively. If you use an AES key, this key cannot be changed once you permanently lock the device.

4. To use the Permanent FlashLock™ feature, select **Lock for both writing and verifying** for FPGA Array and **Lock for both reading and writing** for FlashROM and select the **Permanently lock the security settings** checkbox as shown in the figure below. This will make your device one-time-programmable.

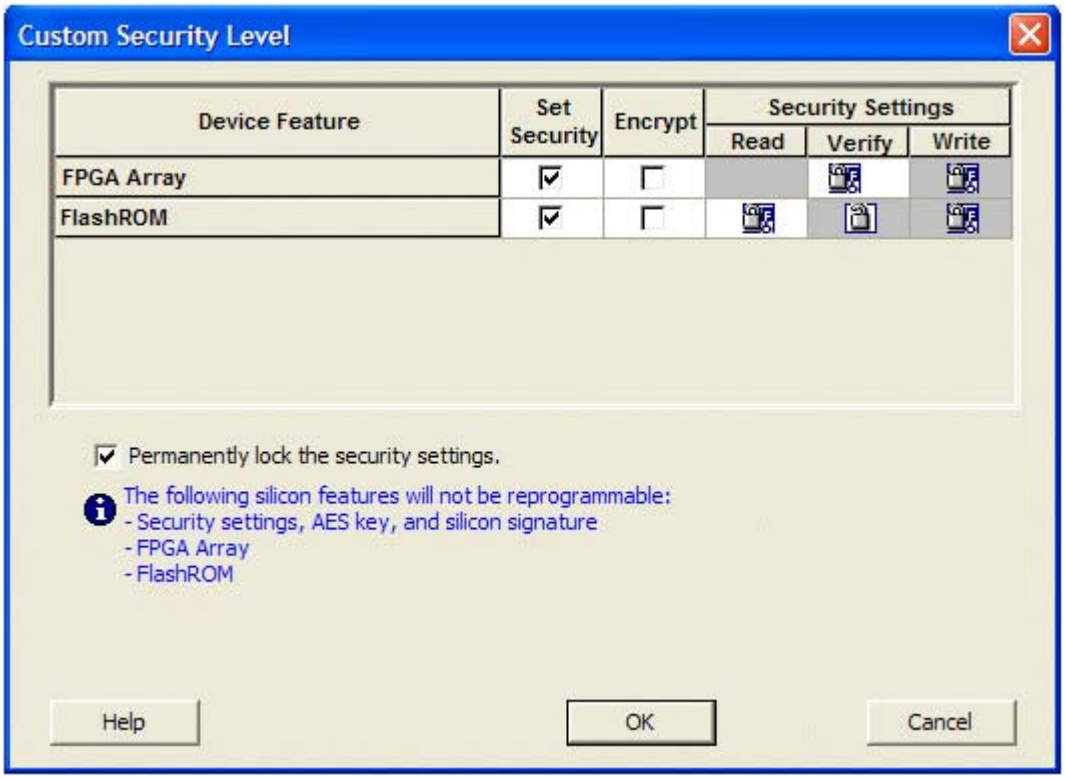


Figure 65 · Custom Security Level

5. Click the **OK** button.

The **Security Settings** page appears with the **Custom security setting** information as shown in the figure below.

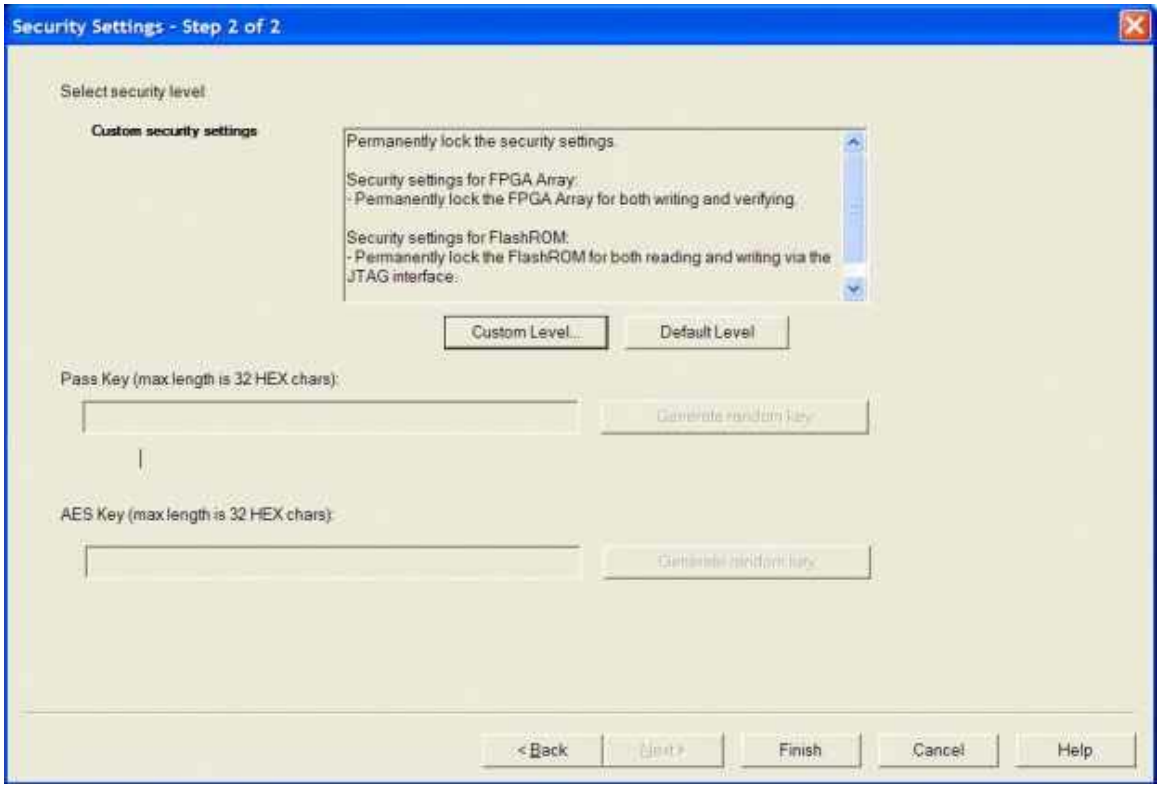


Figure 66 · Security Settings

Custom Serialization Data for FlashROM Region

FlashPoint enables you to specify a custom serialization file as a source to provide content for programming into a Read from file FlashROM region. You can use this feature for serializing the target device with a custom serialization scheme.

To specify a FlashROM region:

1. From the Properties section in the FlashROM Settings page, select the file name of the custom serialization file (see figure below). For more information on custom serialization files, see Custom Serialization Data File Format.

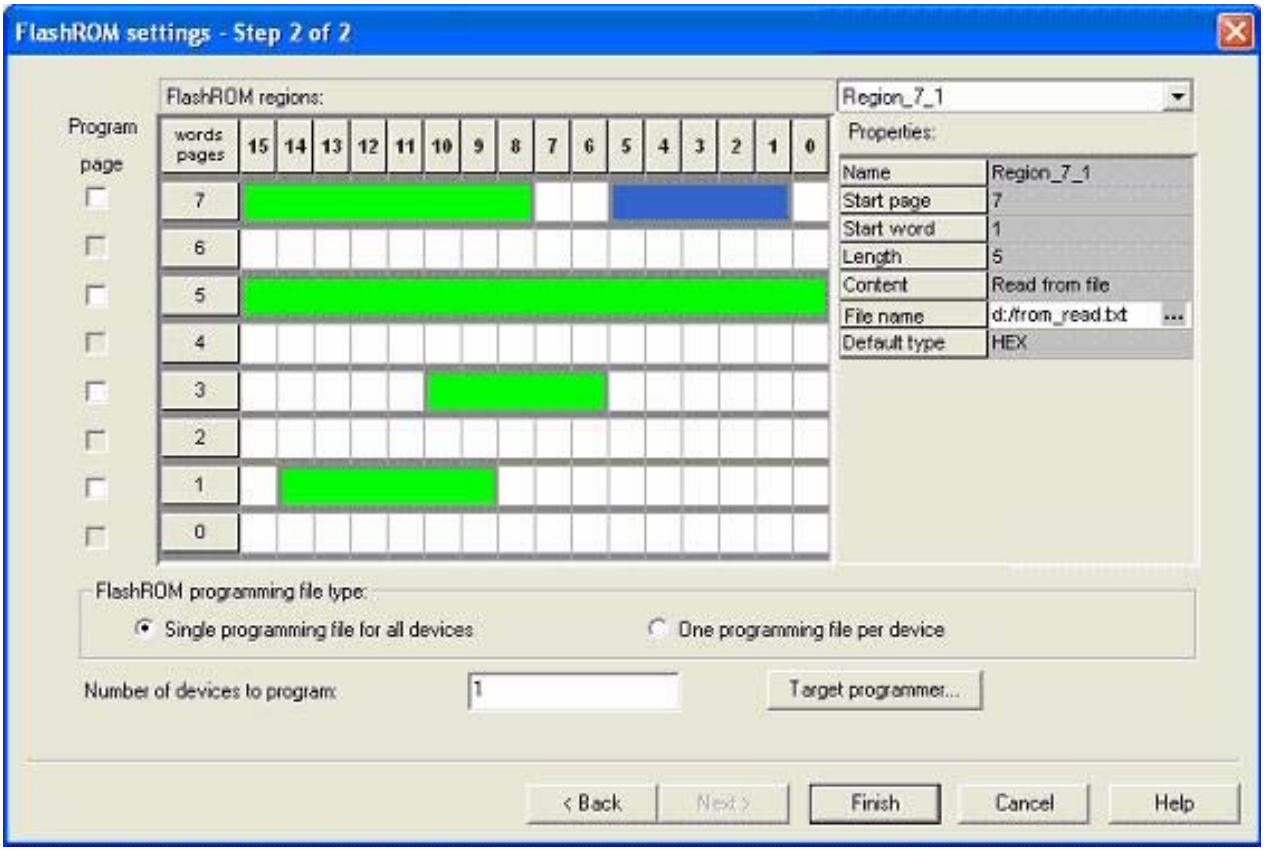


Figure 67 · FlashROM Settings

2. Select the FlashROM programming file type you want to generate from the two options below:
   - Single programming file for all devices option: generates one programming file with all the values in the custom serialization file.
   - One programming file per device: generates one programming file for each value in the custom serialization file.
3. Enter the number of devices you want to program.
4. Click the **Target Programmer** button.
5. Select your target Programmer type.
6. Click **OK**.

Custom Serialization Data File Format

FlashPoint supports custom serialization data files that specify the data in binary, HEX, decimal, or ASCII text. The custom serialization data files may contain multiple data with the Line Feed (LF) character as the delimiter. You can create a file by entering serialization data into any type of text editor. Depending on the serialization data format (hex, ASCII, binary, decimal), input the serialization data according to the size of the region you specified in the FlashROM settings page.

Semantics

Each custom serialization file has only one type of data format (binary, decimal, Hex or ASCII text). For example, if a file contains two different data formats (i.e. binary and decimal) it is considered an invalid file.

The length of each data file must be shorter or equal to the selected region length. If the data is shorter then the selected region length, the most significant bits shall be padded with 0's. If the specified region length is longer then the selected region length, it is considered an invalid file.

The digit / character length is as follows:

```
-Binary digit: 1 bit
-Decimal digit: 4 bits
-Hex digit: 4 bits
-ASCII Character: 8 bits
```

Note the standard example below:

If you wanted to use, for example, device serialization for three devices with serialization data 123, 321, and 456, you would create file name from_read.txt. Each line in from_read.txt corresponds to the serialization data that will be programmed on each device. For example, the first line corresponds to the first device to be programmed, the second line corresponds to the second device to be programmed, and so on.

Hex Serialization Data File Example

The following example is a Hex serialization data file for a 40-bit region. Enter the serialization data below into file created by any text editor:

```
123AEd210
AeB1
0001242E
```

Note: If you enter an invalid Hex digit such as 235SedF1, an error occurs. An error will also occur if you enter data that is out of range, i.e. 4300124EFE.

The following is an example of programming "AeB1" into Region_7_1 located on page 7, from Word 5 to Word 1 in the FlashROM settings page. See Custom serialization data for FlashROM region for more information.

	Table 15	...		Word 5	Word 4	Word 3	Word 2	Word 1	Word 0
Page 7	...	...	...	00	00	00	AE	B1	...

Binary Serialization Data File Example

The following example is a binary serialization data file for a 16-bit region:

1100110011010001

```
100110011010011
11001100110101111 (This is an error: data out of range)
1001100110110111
1001100110110112 (This is an error: invalid binary digit)
```

Decimal Serialization Data File Example

The following example is a decimal serialization data file for a 16-bit region:

```
65534
65535
65536 (This is an error: data out of range)
6553A (This is an error: invalid decimal digit)
```

Text Serialization Data File Example

The following example is a text serialization data file for a 32-bit region:

```
AESB
A )e
ASE3 23 (This is an error: data out of range)
65A~
1234
AEbF
```

Syntax

```
Custom serialization data file
hex region data list> | <decimal region data list> |
 <binary region data list> | <ascii text data list>
Hex region data list = hex data> <new line> { < hex data> <new line> }
Decimal region data list = <decimal data> <new line> {<decimal data><new line> }
Binary region data list = <binary data> <new line> { <binary data> <new line> }
ASCII text region data list = < ascii text data> <new line>
{ < ascii text data> <new line> }
hex data = <hex digit> {<hex digit>}
decimal data = < decimal digit> {< decimal digit>}
binary data = < binary digit> {< binary digit>}
ASCII text data = <ascii character> {< ascii character >}
```

```
new line = LF
binary digit = '0'|'1'
decimal digit = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'| '9'
hex digit = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'|'A'|'B'|'C'|'D' | 'E'| 'F' |
'a'| 'b' | 'c'| 'd' | 'e'| 'f'
ascii character = characters from SP(0x20) to'~'(0x7E).
```

FlashLock

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

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

No Lock

Creates a programming file which does not secure your device.

Permanent Lock

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

Keyed Lock

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

Table 9 · 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

Table 10 · 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

Device	Key Size (bits)	Key Size (hex)
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 STAPL file for IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, and ProASIC devices, or a bitstream file for ProASIC and ProASICPLUS families. 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, select **Programming File** or click the **Programming File** button in the Design Flow window.
2. Select **Bitstream** or **STAPL** from the **File Type** drop-down list box. Bitstream files are not available for IGLOO, Fusion, and ProASIC3 devices.
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, select **Programming File** or click the **Programming File** button in the Design Flow window.
2. **File Type**. Select the appropriate file type in the **File Type** drop-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). Select box to disable clamping diode.
- **Use the JTAG Reset Pull-up Resistor**: (Axcelerator family) Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).
- **Use the Global Set Fuse**: (Axcelerator family) Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option is used, all flip-flops are set to '1' at power up.

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

Generating Prototype Files

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

Note: Designer maps the pins from the RT package to the commercial prototyping vehicle.

To generate prototype files:

1. From the Designer **Tools** menu, choose **Generate Prototype**. This displays the Generate Prototype Files dialog box.
2. Set your **Prototype Die/Package**. The adapter socket required for prototyping varies depending on your target die/package.

   For CQFB to FBGA, see the CQFB to FBGA Adapter Sockets application note.

   For CCGA to FBGA, see the CCGA to FBGA Adapter Sockets application note.
3. Set your Silicon Signature (optional) - Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are '0' through '9', and 'a' through 'f'.
4. Select your **Output directory**. Click the Browse button to navigate to a directory.
5. Set options:

- **Generate probe file also** - This option automatically generates a PRB file for use with Silicon Explorer
- **Use the JTAG reset pull-up resistor** - Select to enable pull-up resistors on the TRSTB pin (JTAG Reset pin which is active low). This is not part of the JTAG standard but can be useful if you want to make sure that the JTAG tap controller is not reset by mistake if the TRSTB pin is not connected. The pull-up resistor guarantees that if the pin is not driven to low (active), the pin is left in an inactive state (high).
- **Use the Global Set Fuse** - Select to set flip-flops to a known state after power-up. If not selected all flip-flops are set to '0' at power up. If this option is used, all flip-flops are set to '1' at power up.

6. Click **OK**. Designer runs through the process of creating the prototype files and generates the AFM for the selected die/package. After the prototype flow is complete, Designer automatically returns to the RT project. You must open your new prototyping ADB file manually to verify it.

TCL Command Reference

Introduction to Tcl Scripting

Tcl, the Tool Command Language, pronounced *tickle*, is an easy-to-learn scripting language that is compatible with Designer software. You can run scripts from either the Windows or UNIX command line or store and run a series of commands in a ".tcl" batch file.

This section provides a quick overview of the main features of Tcl:

- Basic syntax
- Types of Tcl commands
- Variables
- Command substitution
- Quotes and braces
- Lists and arrays
- Control structures
- Handling exceptions
- Print statement and Return values
- Running Tcl scripts from the command line
- Running Tcl scripts from within Designer
- Exporting Tcl scripts
- Extended_run_gui
- Extended_run_shell
- Sample Tcl scripts

For complete information on Tcl scripting, refer to one of the books available on this subject. You can also find information about Tcl at web sites such as http://www.tcl.tk.

Basic Syntax

Tcl scripts contain one or more commands separated by either new lines or semicolons. A Tcl command consists of the name of the command followed by one or more arguments. The format of a Tcl command is:

```
command arg1 ... argN
```

The command in the following example computes the sum of 2 plus 2 and returns the result, 4.

```
expr 2 + 2
```

The **expr** command handles its arguments as an arithmetic expression, computing and returning the result as a string. All Tcl commands return results. If a command has no result to return, it returns an empty string.

To continue a command on another line, enter a backslash (\) character at the end of the line. For example, the following Tcl command appears on two lines:

```
import -format "edif" -netlist_naming "Generic" -edif_flavor "GENERIC" {prepi.edn}
```

Comments must be preceded by a hash character (#). The comment delimiter (#) must be the first character on a line or the first character following a semicolon, which also indicates the start of a new line. To create a multi-line comment, you must put a hash character (#) at the beginning of each line.

Note: Be sure that the previous line does not end with a continuation character (\). Otherwise, the comment line following it will be ignored.

Special Characters

Square brackets ([ ]) are special characters in Tcl. To use square brackets in names such as port names, you must either enclose the entire port name in curly braces, for example, pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High, or lead the square brackets with a slash (\) character as shown in the following example:
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High

Sample Tcl Script

```
#Set up a new design
new_design -name "multiclk" -family "Axcelerator" -path {.}

# Set device, package, speed grade, default I/O standard and
# operating conditions
set_device -die "AX1000" -package "BG729" -speed "-3" \
-voltage "1.5" -iostd "LVTTL" -temprange "COM" -voltrange "COM"

# Import the netlist
import -format "verilog" {multiclk.v}

# Compile the netlist
compile

# Import a PDC file
import_aux -format "pdc" {multiclk.pdc}
```

```
# Run standard layout
layout -incremental "OFF"

# Generate backannotated sdf and netlist file
backannotate -name {multiclk_ba} -format "sdf" -language "Verilog"

# Generate timing report
report -type "timing" -sortby "actual" -maxpaths "100" {report_timing.txt}

# Generate programming file
export -format "AFM" -signature "ffff" {multiclk.afm}
```

Types of Tcl Commands

There are three types of Tcl commands:

- Built-in commands
- Procedures created with the proc command
- Commands built into the Designer software

Built-In Commands

Built-in commands are provided by the Tcl interpreter. They are available in all Tcl applications. Here are some examples of built-in Tcl commands:

- Tcl provides several commands for manipulating file names, reading and writing file attributes, copying files, deleting files, creating directories, and so on.
- exec - run an external program. Its return value is the output (on stdout) from the program, for example:

```
set tmp [ exec myprog ]
puts stdout $tmp
```

- You can easily create collections of values (lists) and manipulate them in a variety of ways.
- You can create arrays - structured values consisting of name-value pairs with arbitrary string values for the names and values.
- You can manipulate the time and date variables.
- You can write scripts that can wait for certain events to occur, such as an elapsed time or the availability of input data on a network socket.

Procedures Created with the Proc Command

You use the proc command to declare a procedure. You can then use the name of the procedure as a Tcl command.

The following sample script consists of a single command named **proc**. The proc command takes three arguments:

- The name of a procedure (myproc)

- A list of argument names (arg1 arg2)
- The body of the procedure, which is a Tcl script

```
proc myproc { arg1 arg2 } {
# procedure body
}
myproc a b
```

Commands Built into the Designer Software

Many functions that you can perform through the Designer software's GUI interface, you can also perform using an equivalent Tcl command. For example, the `backannotate` command is equivalent to executing the Back-Annotate command from Designer's Tools menu. For a list of Tcl commands supported in the Designer software, see "Tcl Commands."

Variables

With Tcl scripting, you can store a value in a variable for later use. You use the set command to assign variables.For example, the following set command creates a variable named x and sets its initial value to 10.

```
set x 10
```

A variable can be a letter, a digit, an underscore, or any combination of letters, digits, and underscore characters. All variable values are stored as strings.

In the Tcl language, you do not declare variables or their types. Any variable can hold any value. Use the dollar sign ($) to obtain the value of a variable, for example:

```
set a 1
set b $a
set cmd expr
set x 11
$cmd $x*$x
```

The dollar sign $ tells Tcl to handle the letters and digits following it as a variable name and to substitute the variable name with its value.

Global Variables

Variables can be declared global in scope using the Tcl global command. All procedures, including the declaration can access and modify global variables, for example:

```
global myvar
```

Command Substitution

By using square brackets ([]), you can substitute the result of one command as an argument to a subsequent command, as shown in the following example:

```
set a 12
set b [expr $a*4]
```

Tcl handles everything between square brackets as a nested Tcl command. Tcl evaluates the nested command and substitutes its result in place of the bracketed text. In the example above, the argument that appears in square brackets in the second set command is equal to 48 (that is, 12* 4 = 48).

Conceptually,

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

expands to

```
set b [expr 12 * 4 ]
```

and then to

```
set b 48
```

Quotes and Braces

The distinction between braces ({ }) and quotes (" ") is significant when the list contains references to variables. When references are enclosed in quotes, they are substituted with values. However, when references are enclosed in braces, they are not substituted with values.

Table 11 · Example

With Braces	With Double Quotes
set b 2	set b 2
set t { 1 $b 3 }	set t " 1 $b 3 "
set s { [expr $b + $b] }	set s " [expr $b + $b] "
puts stdout $t	puts stdout $t
puts stdout $s	puts stdout $s

will output

```
1 $b 3                    vs.        1 2 3
[ expr $b + $b ]                     4
```

FileNames

In Tcl syntax, filenames should be enclosed in braces { } to avoid backslash substitution and white space separation. Backslashes are used to separate folder names in Windows-based filenames. The problem is that sequences of “\n” or “\t” are interpreted specially. Using the braces disables this special interpretation and specifies that the Tcl interpreter handle the enclosed string literally. Alternatively, double-backslash “\\n” and “\\t” would work as well as forward slash directory separators “/n” and “/t”.For example, to specify a file on your Windows PC at c:\newfiles\thisfile.adb, use one of the following:

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

```
C:/newfiles/thisfile.adb
"C:/newfiles/thisfile.adb"
```

If there is white space in the filename path, you must use either the braces or double-quotes. For example:

```
C:\program data\thisfile.adb
```

should be referenced in Tcl script as

```
{C:\program data\thisfile.adb} or "C:\\program data\\thisfile.adb"
```

If you are using variables, you cannot use braces { } because, by default, the braces turn off all special interpretation, including the dollar sign character. Instead, use either double-backslashes or forward slashes with double quotes. For example:

```
"$design_name.adb"
```

Note: To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket.

The following example shows a port name enclosed with curly braces:

```
pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High
```

The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

Lists and Arrays

A list is a way to group data and handle the group as a single entity. To define a list, use curly braces { } and double quotes " ". For example, the following set command {1 2 3 }, when followed by the list command, creates a list stored in the variable "a." This list will contain the items "1," "2," and "3."

```
set a { 1 2 3 }
```

Here's another example:

```
set e 2
set f 3
set a [ list b c d [ expr $e + $f ] ]
puts $a
```

displays (or outputs):

```
b c d 5
```

Tcl supports many other list-related commands such as lindex, linsert, llength, lrange, and lappend. For more information, refer to one of the books or web sites available on this subject.

Arrays

An array is another way to group data. Arrays are collections of items stored in variables. Each item has a unique address that you use to access it. You do not need to declare them nor specify their size.

Array elements are handled in the same way as other Tcl variables. You create them with the set command, and you can use the dollar sign ($) for their values.

```
set myarray(0) "Zero"
set myarray(1) "One"
```


```
set myarray(2) "Two"
for {set i 0} {$i < 3} {incr i 1} {
```

Output:

```
Zero
One
Two
```

In the example above, an array called "myarray" is created by the set statement that assigns a value to its first element. The for-loop statement prints out the value stored in each element of the array.

Special Arguments (command-line parameters)

You can determine the name of the Tcl script file while executing the Tcl script by referring to the $argv0 variable.

```
puts "Executing file $argv0"
```

To access other arguments from the command line, you can use the `lindex` command and the *`argv`* variable:

To read the the Tcl file name:

```
lindex $argv 0
```

To read the first passed argument:

```
lindex $argv 1
```

Example

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

Control Structures

Tcl control structures are commands that change the flow of execution through a script. These control structures include commands for conditional execution (if-then-elseif-else) and looping (while, for, catch).

An "if" statement only executes the body of the statement (enclosed between curly braces) if the Boolean condition is found to be true.

If/Else Statements

```
if { "$name" == "paul" } then {
…
# body if name is paul
} elseif { $code == 0 } then {
…
# body if name is not paul and if value of variable code is zero
} else {
…
```

```
# body if above conditions is not true
}
```

For Loop Statement

A "for" statement will repeatedly execute the body of the code as long as the index is within a specified limit.

```
for { set i 0 } { $i < 5 } { incr i } {
…
# body here
}
```

While Loop Statement

A "while" statement will repeatedly execute the body of the code (enclosed between the curly braces) as long as the Boolean condition is found to be true.

```
while { $p > 0 } {
…
}
```

Catch Statement

A "catch" statement suspends normal error handling on the enclosed Tcl command. If a variable name is also used, then the return value of the enclosed Tcl command is stored in the variable.

```
catch { open "$inputFile" r } myresult
```

Handling Exceptions (Tcl scripting)

To control the flow of the Designer software based on certain conditions (for example, success or failure of certain commands), you can use the Tcl built-in catch command as follows:

```
if { [ catch {open_design $des_name.adb} ] } {
    puts "Cannot open $des_name.adb"
    export -format "log" -diagnostic $des_name.log"
   exit 1
    } else {
   puts "Design $des_name.adb Successfully Opened"
}
## set layout mode to standard
layout -incremental "OFF"
if { [ catch {layout} ] } {
    puts "Layout Failed"
    export -format "log" -diagnostic $des_name.log"
    exit 1
} else {
   puts "layout successful"
  export -format log "$des_name.log"
  save_design "$des_name.adb";
  close_design
```

```
}
```

Print Statement and Return Values

Print Statement

Use the puts command to write a string to an output channel. Predefined output channels are “stdout” and “stderr.” If you do not specify a channel, then puts will display text to the stdout channel.

Example:

```
set a [ myprog arg1 arg2 ]
puts "the answer from myprog was $a (this text is on stdout)"
puts stdout “this text also is on stdout”
```

Return Values

The return code of a Tcl command is a string. You can use a return value as an argument to another function by enclosing the command with square brackets [ ].

Example:

```
set a [ prog arg1 arg2 ]
exec $a
```

The Tcl command “exec” will run an external program. The return value of “exec” is the output (on stdout) from the program.

```
Example:
set tmp [ exec myprog ]
puts stdout $tmp
```

Running Tcl Scripts from the Command Line

You can run Tcl scripts from your Windows or Unix command line as well as pass arguments to scripts from the command line.

To execute a Tcl script file in the Designer software from a shell command line:

At the prompt, type the path to the Actel software followed by the word "SCRIPT" and a colon, and then the name of the script file as follows:

```
<location of Actel software>\bin\designer SCRIPT:<filename>
```

where `<location of Actel software>` is the root directory in which you installed the Actel software, and`<filename>`is the name, including a relative or full path, of the Tcl script file to execute.

```
For example, to run the Tcl script file named "myscript.tcl" from the command line, you can type:
C:\libero\designer\bin\designer SCRIPT:myscript.tcl
```

If `myscript.tcl` is in a particular folder named "mydesign", you can use SCRIPT_DIR to change the current working directory before calling the script, as in the following example:

```
C:\libero\designer\bin\designer SCRIPT:myscript.tcl "SCRIPT_DIR:C:\actelprj\mydesign"
```

To pass arguments from the command line to your Tcl script file:

At the prompt, type the path to the Actel software followed by the SCRIPT argument. Enclose the entire argument expression in double quotes:

```
<location of Actel software>\bin\designer "SCRIPT:<filename arg1 arg2 ...>"
```

where `<location of Actel software>` is the root directory in which you installed the Actel software, and `<filename arg1 arg2 ...>`is the name, including a relative or full path, of the Tcl script file and arguments you are passing to the script file.

For example,

```
C:\libero\designer\bin\designer "SCRIPT:myscript.tcl  one two three"
```

To obtain the output from the log file:

At the prompt, type the path to the Actel software followed by the SCRIPT and LOGFILE arguments.

```
<location of Actel software>\bin\designer "SCRIPT:<filename arg1 arg2 ...>"
LOGFILE:<output.log>
```

where `<location of Actel software>` is the root directory in which you installed the Actel software, `<filename arg1 arg2 ...>`is the name, including a relative or full path, of the Tcl script file and arguments you are passing to the script file, and `output.log` is the name of the log file.

For example,

```
C:\libero\designer\bin\designer "SCRIPT:myscript.tcl  one two three" "LOGILE:output.log"
```

Running Tcl Scripts from within Designer

Instead of running scripts from the command line, you can use Designer's Execute Script dialog box to run a script.

To execute a Tcl script file within Designer:

1. From the **File** menu, choose **Execute Script** to display the **Execute Script** dialog box.

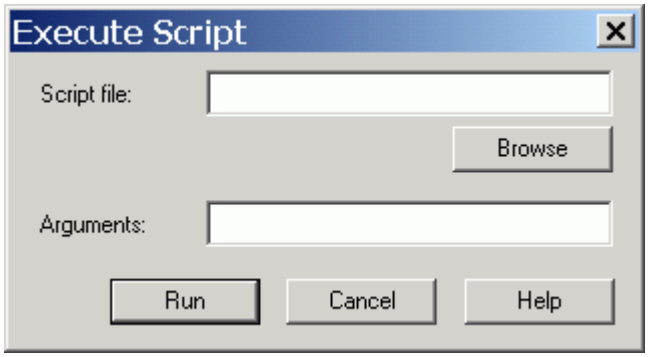


Figure 68 · Execute Script Dialog Box

2. Click **Browse** to display the **Open** dialog box, in which you can navigate to the folder containing the script file to open. When you click **Open**, Designer enters the full path and script filename into the Execute Script dialog box for you.
3. In the Arguments edit box, enter the arguments to pass to your Tcl script as shown in the following sample Execute Script dialog box. Separate each argument by a space character. For information about accessing arguments passed to a Tcl script, see "Running Scripts from the command line."

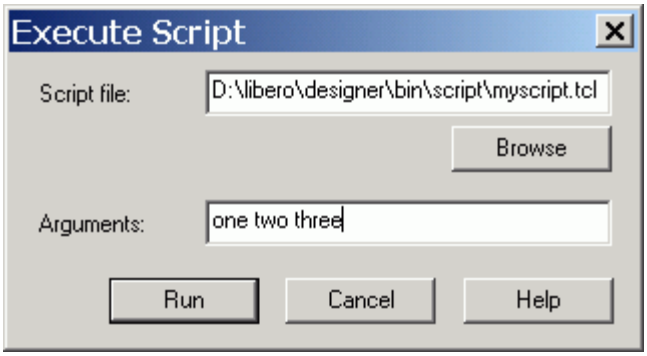


Figure 69 · Execute Script Dialog Box Example

4. Click **Run**.

Specify your arguments in the Execute Script dialog box. To get those argument values from your Tcl script, use the following:

```
puts "Script name: $argv0"
puts "Number of arguments: $argc"
set i 0
foreach arg $argv {
puts "Arg $i : $arg"
incr i
}
```

Exporting Tcl Scripts

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

To export a Tcl session script from Designer:

1. From the **File** menu, choose **Export>Script Files**. The **Export Script Files** dialog box appears.
2. From the **Save in** drop-down menu, navigate to the folder in which you want to save the script files.
3. Type a filename for this Tcl script file.
4. Click **Save**. The **Script Export Options** dialog box appears

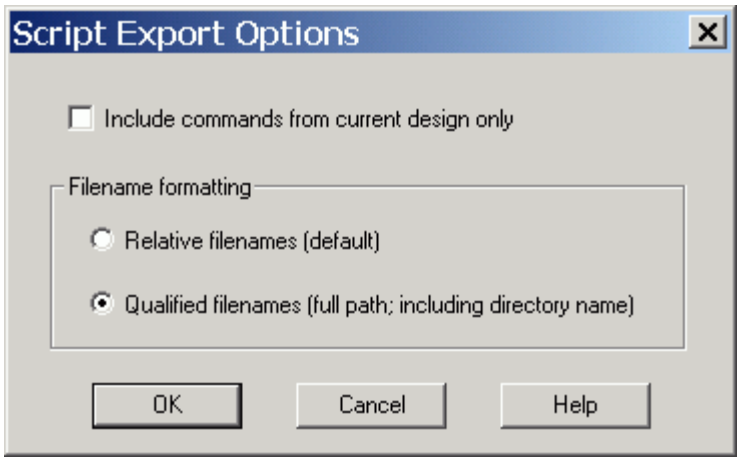


Figure 70 · Script Export Options

5. Check the **Include Commands from Current Design Only** check box if you want to export commands relative to the current design only. This option applies only if you opened more than one design in your current session. If so, and you do not check this box, Designer exports all commands from your current session.
6. Select the radio button for the appropriate filename formatting. To export filenames relative to the current working directory, select **Relative filenames (default)** formatting. To export filenames that include a fully specified path, select **Qualified filenames (full path; including directory name)** formatting.
7. Click **OK**.

The Tcl script file is written to the specified filename.

Note:

- When exporting Tcl scripts, Designer always encloses filenames in curly braces to ensure portability.
- Designer software will not write out any Tcl variables or flow-control statements to the exported Tcl file, even if you had executed the design commands using your own Tcl script. The exported Tcl file will only contain the Designer tool commands and their accompanying arguments.

extended_run_shell

The extended_run_shell Tcl script enables you to run the multiple pass layout in batch mode from a command line. Use this script from the tcl shell "acttclsh". **This is the script or command-line equivalent to using the multiple pass layout in the GUI.**

```
$ACTEL_SW_DIR/bin/acttclsh extended_run_shell.tcl -adb adbFileName.adb [-n numPasses] [-
starting_seed_index numIndex] [-save_all] [-compare_criteria value] [-c clockName] [-
analysis value] [-slack_criteria value] [-timing_driven|-standard] [-stop_on_success] [-
run_placer value] [-place_incremental value] [-route_incremental value] [-effort_level
numLevel] [-timing_weight numWeight] [-placer_high_effort value] [-mindel_repair value] [-
power_driven value]
```

Note: This is not a Tcl command; it is a shell script that can be run from the command line. To invoke multiple pass layout within another Designer Tcl script, refer to extended_run_gui.

Arguments

`-adb adbFileName.adb`

This is the design file to run multiple passes of layout.

`[-n numPasses]`

Sets the number of passes to run. The default number of passes is 5.

`[-starting_seed_index numIndex]`

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Its value should range from 1 to 101. If not specified, the default behavior is to continue from the last seed index which was used.

`[-save_all]`

Saves all intermediate designs in<adbFileName>_r<runNum>_s<seedIndex>.adb. The best result is also stored to the original *.adb file as well. The default behavior does not save all results.

`[-compare_criteria value]`

The following table shows the acceptable values for this argument:

Value	Description
frequency	Sets the criteria for comparing results between passes to be clock frequency based. This is the default. This option enables the -c option (described below).
violations	Sets the criteria for comparing results between passes to be timing violations (slack) based. This option enables the -analysis, -slack_criteria, and -stop_on_success options (described below).
power	Sets the criteria for comparing results between passes to be based on the lowest total power.

`[-c clockName]`

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used.

`[-analysis value]`

Applies only when the timing violations comparison criteria is used. The following table shows the acceptable values for this argument:

Value	Description
max	Examines timing violations (slacks) obtained from maximum delay analysis. This is the default.
min	Examines timing violations (slacks) obtained from minimum delay analysis.

`[-slack_criteria value]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slacks) is determined by the -analysis option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.
tns	Sets the timing violations criteria to total negative slack. For each pass obtains the sum of negative slacks from the first 100 paths from the timing violations report. The largest value out of all passes will determine the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass

`[-stop_on_success]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slacks) is determined by the -analysis option. Stops performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

`[-timing_driven|-standard]`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`[-run_placer` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placer. This is the default.
off	Skips placer.

`[-place_incremental` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point for each pass.
fix	Locks previous placement for each pass.

`[-route_incremental` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous routing. This is the default.

Value	Description
on	Sets the previous routing as the initial starting point for each pass.

`[-effort_level numLevel]`

This is an advanced option that is available only for Axcelerator, SX-A, and eX. It specifies the placement effort level.

`[-timing_weight numWeight]`

This is an advanced option that is available only for ProASIC^{PLUS}, ProASIC, SX-A, and eX. It specifies the layout timing weight.

`[-placer_high_effort value]`

This is an advanced option that is available only for IGLOO, Fusion, and ProASIC3 families. The following table shows the acceptable values for this argument:

Value	Description
off	Runs layout in regular effort. This is the default.
on	Activates high effort layout mode.

`[-mindel_repair value]`

This is an advanced option that is available only for IGLOO, Fusion, and ProASIC3 families. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run minimum delay violations repair. This is the default.
on	Enables repair of minimum delay violations during route.

`[-power_driven value]`

This option is available only for IGLOO, Fusion, ProASIC3 and Axceleratorfamilies. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout. This is the default.
on	Enables power-driven layout.

Return

A non-zero value will be returned on error.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, eX, and SX-A

Exceptions

None

Example

1. On *my.adb*, run 5 (default) passes continuing from the last seed index using slowest clock frequency (default) comparison criteria.

```
% acttclsh extended_run_shell.tcl -adb my.adb
```

2. On *my.adb*, run 3 passes starting with seed index 6, saving all results, using clock frequency comparison criteria for clock "PCI_CLK".

```
% acttclsh extended_run_shell.tcl -adb my.adb -n 3 -starting_seed_index 6 -save_all -c PCI_CLK
```

3. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with maximum delay (default) analysis and worst slack (default) criteria; invoke high effort layout.

```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations -placer_high_effort on
```

4. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with maximum delay (default) analysis and total negative slack criteria; invoke placement effort level 5.

```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations -slack_criteria tns -effort_level 5
```

5. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with minimum delay analysis and worst slack (default) criteria; stop if there are no violations.

```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations -analysis min -stop_on_success
```

6. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with minimum delay analysis and total negative slack criteria; invoke repair of minimum delay violations.

```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations -analysis min -slack_criteria tns -mindel_repair on
```

See Also

Running Layout

Multiple Pass Layout

extended_run_gui

extended_run_gui

This script is used to reproduce the GUI behavior and is more suited for running through Designer or inside another Designer TCL script.

```
extended_run_gui.tcl [-n numPasses] [-starting_seed_index numIndex] [-save_all] [-compare_criteria value] [-c clockName] [-analysis value] [-slack_criteria value] [-timing_driven|-standard] [-stop_on_success] [-run_placer value] [-place_incremental value] [-route_incremental value] [-effort_level numLevel] [-timing_weight numWeight] [-placer_high_effort value] [-mindel_repair value] [-power_driven value]
```

The only difference from extended_run_shell Tcl script is that the *extended_run_gui.tcl* script does not need the `-adb` argument and assumes that the design is already saved and open.

To invoke extended_run_gui from Designer:

1. Open an *.adb file in Designer.
2. From the **File** menu, select **Execute Script**. This opens the Execute Script dialog box.

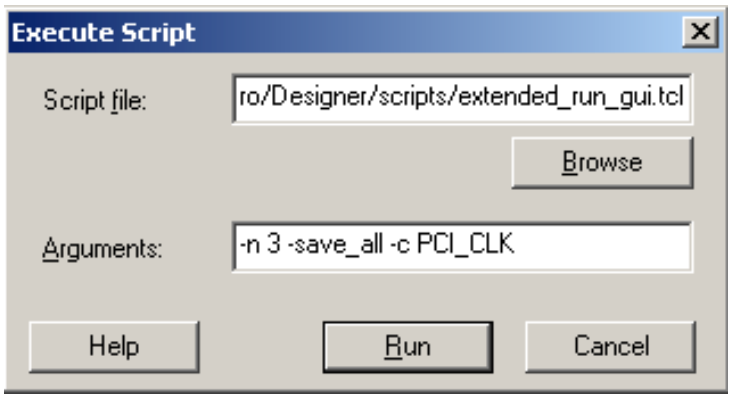


Figure 71 · Execute Script Dialog Box

3. Find the *extended_run-gui.tcl* script under *ACTEL_SW_DIR/scripts* and then copy all the parameters in to the arguments section.
4. Click **Run**.

To invoke extended_run_gui from within a TCL script:

1. Save the design in compiled state. ...

```
compile
save_design "my.adb"
```

2. Override the original argument list in the caller script and then source the *extended_run_gui.tcl* script. `set save_argv0 $::argv0`

```
set save_argv $::argv
set ACTEL_SW_DIR $env(ACTEL_SW_DIR)
set ::argv0 "$ACTEL_SW_DIR/scripts/extended_run_gui.tcl"
set ::argv [list -n 3 -save_all -c PCI_CLK]
set ::argc [llength $::argv]
source $::argv0
set ::argv0 $save_argv0
set ::argv $save_argv
set ::argc [llength $::argv]
```

See Also

Running Layout

Multiple Pass Layout

extended_run_shell

Sample Tcl Script

Basic Design Flow

The following script creates a new design named *prepi* for the SX-A family:

```
#Set up a new design
new_design -name "prepi" -family "SXA"
#set device name, package name
set_device -die "A54SX32A" -package "176 TQFP"
#set device speed and operating conditions
set_device -speed "-1" -temprange "com" -voltrange "com"
#import netlist and pin files
import -format "edif" -netlist_naming "Generic" \
-edif_flavor "GENERIC" {prepi.edn}
import -format "pin" {prepi.pin}
compile
#layout standard mode
layout -incremental "OFF"
#extract sdf file
export -format "sdf" {prepi.sdf}
save_design {prepi.adb}
close_design
```

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

About Designer Tcl Commands

A Tcl (Tool Command Language) file contains scripts for simple or complex tasks. You can run scripts from either the Windows or UNIX command line or store and run a series of Tcl commands in a ".tcl" batch file. You can also run scripts from within Designer.

Designer supports the following Tcl scripting commands:

Command	Action
all_inputs	Returns an object representing all input and inout pins in the current design
all_outputs	Returns an object representing all output and inout pins in the current design
all_registers	Returns an object representing register pins or cells in the current scenario based on the given parameters
are_all_source_files_current	Audits all source files and determines whether or not they are out of date / imported into the workspace

Command	Action
backannotate	Extracts timing delays from your post layout data
check_timing_constraints	Checks all timing constraints in the current timing scenario for validity
clone_scenario	Creates a new timing scenario by duplicating an existing one
close_design	Closes the current design
compile	Performs design rule check and optimizes the input netlist before translating the source code into machine code
create_clock	Creates a clock constraint on the specified ports/pins, or a virtual clock if no source is specified
create_generated_clock	Creates an internally generated clock constraint on the ports/pins and defines its characteristics
create_scenario	Creates a new timing scenario with the specified name
delete_scenario	Deletes the specified timing scenario
export	Converts a file from its current format into the specified file format, usually for use in another program
extended_run_shell	Runs multiple iterations of layout through Designer
get_cells	Returns an object representing the cells (instances) that match those specified in the pattern argument
get_clocks	Returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario
get_current_scenario	Returns the name of the current timing scenario
get_defvar	Returns the value of the Designer internal variable you specify
get_design_filename	Returns the fully qualified path of the specified design file
get_design_info	Returns detailed information about your design, depending on which arguments you specify
get_nets	Returns an object representing the nets that match those specified in the pattern argument

Command	Action
get_out_of_date_files	Audits all files returns a list of filenames that are out of date
get_pins	Returns an object representing the pin(s) that match those specified in the pattern argument
get_ports	Returns an object representing the port(s) that match those specified in the pattern argument
import_aux	Imports the specified file as an auxiliary file, which are not audited and do not require you to re-compile the design
import_source	Imports the specified file as a source file, which include your netlist and design constraints
is_design_loaded	Returns True if the design is loaded into Designer; otherwise, returns False
is_design_modified	Returns True if the design has been modified since it was last compiled; otherwise, returns False
is_design_state_complete	Returns True if the specified design state is complete (for example, you can inquire as to whether a die and package has been selected for the design); otherwise, returns False
is_source_file_current	Audits the source file and determines whether or not the file is out of date / imported into the workspace
layout	Place-and-route your design
layout (advanced options for the SX family)	Sets advanced place-and-route features for SX family designs
layout - Axcelerator	Sets advanced place-and-route features for Axcelerator family designs
list_clocks	Returns details about all of the clock constraints in the current timing constraint scenario
list_clock_latencies	Returns details about all of the clock latencies in the current timing constraint scenario
list_disable_timings	Returns the list of disable timing constraints for the current scenario
list_false_paths	Returns details about all of the false paths in the current timing constraint scenario

Command	Action
list_generated_clocks	Returns details about all of the generated clock constraints in the current timing constraint scenario
list_input_delays	Returns details about all of the input delay constraints in the current timing constraint scenario
list_max_delays	Returns details about all of the maximum delay constraints in the current timing constraint scenario
list_min_delays	Returns details about all of the minimum delay constraints in the current timing constraint scenario
list_multicycle_paths	Returns details about all of the multicycle paths in the current timing constraint scenario
list_objects	Returns a list of names of the objects in the specified list
list_output_delays	Returns details about all of the output delay constraints in the current timing constraint scenario
list_scenarios	Returns a list of names of all of the available timing scenarios
new_design	Creates a new design (.adb) file in a specific location for a particular design family such as Axcelerator or ProASIC3
open_design	Opens an existing design in the Designer software
pin_assign	Assigns the named pin to the specified port but does not lock its assignment.
pin_commit	Saves the pin assignments to the design (.adb) file.
pin_fix	Locks the pin assignment for the specified port, so the pin cannot be moved during place-and-route.
pin_fix_all	Locks all the assigned pins on the device so they cannot be moved during place-and-route.
pin_unassign	Unassigns a specific pin from a specific port. The unassigned pin location is then available for other ports.
pin_unassign_all	Unassigns all pins from a specific port.
pin_unfix	Unlocks the specified pin from its port.

Command	Action
layout (advanced options for ProASIC	Sets advanced place-and-route features for ProASIC family designs
remove_clock	Removes the specified clock constraint from the current timing scenario
remove_clock_latency	Removes a clock source latency from the specified clock and from all edges of the clock
remove_disable_timing	Removes a disable timing constraint by specifying its arguments, or its ID
remove_false_path	Removes a false path from the current timing scenario by specifying either its exact arguments or its ID
remove_generated_clock	Removes the specified generated clock constraint from the current scenario
remove_input_delay	Removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove
remove_max_delay	Removes a maximum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.
remove_min_delay	Removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID
remove_multicycle_path	Removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID
remove_output_delay	Removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove
rename_scenario	Renames the specified timing scenario with the new name provided
report	Generates the type of report you specify: Status, Timing, Timer Violations, Flip-flop, Power, Pin, or I/O Bank
report (Activity and Hazards Power Report)	Reads a VCD file and reports transitions and hazards for each clock cycle of the VCD file.
report (Bottleneck) using SmartTime	Creates a bottleneck report
report (Cycle Accurate Power Report)	Reports a power waveform with one power value per clock period or half-period instead of an average power for the whole simulation

Command	Action
Report (Data History)	Reports new features and enhancements, bug fixes and known issues for the current release that may impact the power consumption of the design
report (Datasheet) using SmartTime	Creates a datasheet report
Report (Power)	Creates a Power report, which enables you to determine if you have any power consumption problems in your design
Report (Power Scenario)	Creates a scenario power report, which enables you to enter a duration for a sequence of previously defined power modes and calculate the average power consumption and the excepted battery life for this sequence.
report (Timing) using SmartTime	Creates a timing report
report (Timing violations) using SmartTime	Creates a timing violations report
set_clock_latency	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
set_current_scenario	Specifies the timing scenario for the Timing Analyzer to use
save_design	Writes the design to the specified filename
set_defvar	Sets the value of the Designer internal variable you specify >
set_design	Specifies the design name, family and path in which Designer will process the design
set_device	Specifies the type of device and its parameters
set_disable_timing	Disables timing arcs within a cell and returns the ID of the created constraint
set_false_path	Identifies paths that are considered false and excluded from the timing analysis in the current timing scenario
set_input_delay	Creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario
set_max_delay	Specifies the maximum delay for the timing paths in the current scenario
set_min_delay	Specifies the minimum delay for the timing paths in the current scenario
set_multicycle_path	Defines a path that takes multiple clock cycles in the current scenario

Command	Action
set_output_delay	Defines the output delay of an output relative to a clock in the current scenario
smartpower_add_new_custom_mode	Creates a new custom mode
smartpower_add_new_scenario	Creates a new scenario
smartpower_add_pin_in_domain	Adds a pin to either a Clock or Set domain
smartpower_commit	Saves the changes made in SmartPower to the design file (.adb) in Designer
smartpower_create_domain	Creates a new clock or set domain
smartpower_edit_custom_mode	Edits a custom mode
smartpower_edit_scenario	Edits a scenario
smartpower_initialize_allclocks	Initializes the clock frequency and the data frequency of all clock domains with the initialization options
smartpower_initialize_clock_with_constraints	Initializes the clock frequency and the data frequency of a single clock domain with a specified clock name and the initialization options.
smartpower_remove_custom_mode	Removes a custom mode
smartpower_remove_domain	Removes an existing domain
smartpower_remove_file	Removes a VCD file from the specified mode
smartpower_remove_pin_enable_rate	Enables you to annotate the enable rate value of a pin driving an enable pin
smartpower_remove_pin_frequency	Removes the frequency of an existing pin
smartpower_remove_pin_of_domain	Removes a clock pin or a data pin from a Clock or Set domain, respectively.
smartpower_remove_scenario	Removes a scenario from the current design
smartpower_remove_vcd	Removes an existing VCD file from a mode or entire design
smartpower_restore	Restores previously committed constraints
smartpower_set_battery_capacity	Sets the battery capacity
smartpower_set_cooling	Sets the cooling style to one of the predefined types, or a custom value

Command	Action
smartpower_set_default_enable_rate	Sets the enable-rate of one of the enable set domains: IOEnable_Set or MemoriesEnableSet
smartpower_set_domain_frequency	Sets the frequency of a domain
smartpower_set_mode_for_analysis	Sets the mode for cycle-accurate power analysis
smartpower_set_operating_condition	Sets the operating conditions used in SmartPower to best, typical, or worst case
smartpower_set_pin_enable_rate	Enables you to annotate the enable rate value of a pin driving an enable pin
smartpower_set_pin_frequency	Sets the frequency of an existing pin
smartpower_set_preferences	Sets SmartPower preferences such as power unit, frequency unit, operating mode, operating conditions, and toggle
smartpower_set_scenario_for_analysis	Sets the scenario for cycle-accurate power analysis
smartpower_set_temperature_opcond	Sets the temperature in the operating conditions used in SmartPower
smartpower_set_thermalmode	Sets the mode of computing junction temperature
smartpower_set_voltage_opcond	Sets the voltage in the operating conditions used in SmartPower
smartpower_temperature_opcond_set_design_wide	Sets the temperature for SmartPower design-wide operating conditions
smartpower_temperature_opcond_set_mode_specific	Sets the temperature for SmartPower mode-specific operating conditions
smartpower_voltage_opcond_set_design_wide	Sets the voltage settings for SmartPower design-wide operating conditions
smartpower_voltage_opcond_set_mode_specific	Sets the voltage settings for SmartPower mode-specific use operating conditions
st_create_set	Creates a set of paths to be analyzed
st_commit	Saves the changes made in SmartTime to the design (.adb) file
st_edit_set	Modifies the paths in a user set

Command	Action
st_expand_path	Displays expanded path information (path details) for paths
st_list_paths	Displays the list of paths in the same tabular format shown in SmartTime
st_remove_set	Deletes a user set from the design
st_restore	Restores constraints previously committed in SmartTime
st_set_options	Sets options for timing analysis
timer_add_clock_exception	Adds an exception to or from a pin with respect to a specified clock
timer_add_pass	Adds the pin to the list of pins for which the path must be shown passing through in the timer
timer_add_stop	Adds the specified pin to the list of pins through which the paths will not be displayed in the timer
timer_commit	Saves the changes made to constraints in Timer into the Designer database.
timer_get_path	Displays the Timer path information in the Log window
timer_get_clock_actuals	Displays the actual clock frequency in the Log window
timer_get_clock_constraints	Displays the clock constraints (period/frequency and dutycycle) in the Log window
timer_get_maxdelay	Displays the maximum delay constraint between two pins of a path in the Log window
timer_get_path_constraints	Displays the path constraints set for maxdelay in the Timer in the Log window
timer_remove_clock_exception	Removes the previously set clock constraint
timer_remove_pass	Removes the previously entered path pass constraint
timer_remove_stop	Removes the path stop constraint on the specified pin
timer_restore	Restores previously committed constraints
timer_setenv_clock_freq	Sets a required clock frequency, in MHz, for the specified clock
timer_setenv_clock_period	Sets the clock period constraint for the specified clock

Command	Action
timer_set_maxdelay	Adds a maximum delay constraint for the path
timer_remove_all_constraints	Removes all the timing constraints previously entered in the Designer system

Note: Tcl commands are case sensitive. However, their arguments are not.

See Also

Introduction to Tcl scripting

Basic syntax

Tcl Command Documentation Conventions

The following table shows the typographical conventions used for the Tcl command syntax.

Syntax Notation	Description
`command -argument`	Commands and arguments appear in Courier New typeface.
`variable`	Variables appear in blue, italic Courier New typeface. You must substitute an appropriate value for the variable.
[`-argument`*`value`*] [*`variable`*]+	Optional arguments begin and end with a square bracket with one exception: if the square bracket is followed by a plus sign (+), then users must specify at least one argument. The plus sign (+) indicates that items within the square brackets can be repeated. Do not enter the plus sign character.

Note: All Tcl commands are case sensitive. However, their arguments are not.

Examples

Syntax for the get_defvar command followed by a sample command:

```
get_defvar variable
```

```
get_defvar "DESIGN"
```

Syntax for the backannotate command followed by a sample command:

```
backannotate -name file_name -format format_type -language language -dir directory_name [-
netlist] [-pin]
```

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

Wildcard Characters

You can use the following wildcard characters in names used in Tcl commands:

Wildcard	What it Does
\	Interprets the next character literally
?	Matches any single character
*	Matches any string
[]	Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range)

Note: The matching function requires that you add a slash (\) before each slash in the port, instance, or net name when using wildcards in a PDC command and when using wildcards in the Find feature of the MultiView Navigator. For example, if you have an instance named "A/B12" in the netlist, and you enter that name as "A\\/B*" in a PDC command, you will not be able to find it. In this case, you must specify the name as A\\\\/B*.

Special Characters [ ], { }, and \

Sometimes square brackets ([ ]) are part of the command syntax. In these cases, you must either enclose the open and closed square brackets characters with curly brackets ({ }) or precede the open and closed square brackets ([ ]) characters with a backslash (\). If you do not, you will get an error message.

For example:

```
pin_assign -port {LFSR_OUT[0]} -pin 15
or
pin_assign -port LFSR_OUT\[0\] -pin 180
```

Note: Tcl commands are case sensitive. However, their arguments are not.

Entering Arguments on Separate Lines

To enter an argument on a separate line, you must enter a backslash (\) character at the end of the preceding line of the command as shown in the following example:

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

See Also

Introduction to Tcl scripting

Basic syntax

About Designer Tcl commands

all_inputs

Returns an object representing all input and inout pins in the current design.

```
all_inputs
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

See Also

Tcl documentation conventions

all_outputs

Returns an object representing all output and inout pins in the current design.

```
all_outputs
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

See Also

Tcl documentation conventions

all_registers

Returns an object representing register pins or cells in the current scenario based on the given parameters.

```
all_registers [-clock clock_name]
[-async_pins][-output_pins][-data_pins][-clock_pins]
```

Arguments

-clock *clock_name*

Specifies the name of the clock domain to which the registers belong. If no clock is specified, all registers in the design will be targeted.

-async_pins

Lists all register pins that are async pins for the specified clock (or all registers asynchronous pins in the design).

-output_pins

Lists all register pins that are output pins for the specified clock (or all registers output pins in the design).

-data_pins

Lists all register pins that are data pins for the specified clock (or all registers data pins in the design).

-clock_pins

Lists all register pins that are data pins for the specified clock (or all registers clock pins in the design).

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands:

set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2.000 -from { ff_m:CLK ff_s2:CLK } -to [all_registers -clock_pins -clock {
ff_m:Q }]
```

See Also

Tcl documentation conventions

are_all_source_files_current

Audits all source files and determines whether or not they are out of date / imported into the workspace. Returns '1' if all source files are current Returns '0' if all source files are not current This command ignores the Audit settings in your ADB file.

```
are_all_source_files_current
```

Arguments

None

Supported Family

All

Exceptions

- The command will return an error if arguments are passed.

Example

The following code will determine if your source files are current.

```
are_all_source_files_current
```

See Also

get_out_of_date_files
is_source_file_current

Backannotate

Equivalent to executing the Back-Annotate command from 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.

Actel recommends that you 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.

```
backannotate -name file_name -format format_type -language language-dir directory_name [-
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 Language options are:

Value	Description
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.

`-pin`

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

Supported Families

All

Exceptions

- The -pin argument is not supported for ProASIC and ProASICPLUS families.

Examples

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
```

See Also

Tcl command documentation conventions

check_timing_constraints

Checks all timing constraints in the current timing scenario for validity.

```
check_timing_constraints
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
check_timing_constraints
```

See Also

Tcl documentation conventions

close_design

Closes the current design and brings Designer to a fresh state to work on a new design. This is equivalent to selecting the Close command from the File menu.

```
close_design
```

Arguments

None

Supported Families

All

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

clone_scenario

Creates a new timing scenario by duplicating an existing one. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
clone_scenario name -source origin
```

Arguments

name

Specifies the name of the new timing scenario to create.

-source *origin*

Specifies the source of the timing scenario to clone (copy). The source must be a valid, existing timing scenario.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Description

This command creates a timing scenario with the specified name, which includes a copy of all constraints in the original scenario (specified with the -source parameter). The new scenario is then added to the list of scenarios.

Example

```
clone_scenario scenario_A -source {Primary}
```

See Also

create_scenario
delete_scenario
Tcl documentation conventions

Compile

Performs design rule check on the input netlist. Compile also performs some optimizations on the design through logic combining and buffer tree modifications. Compile options vary by family.

Supported Families

- MX, SX, SX-A, and eX
- Axcelerator
- ProASIC, ProASICPLUS
- IGLOO, Fusion, ProASIC3

See Also

Setting Compile Options

create_clock

Creates a clock constraint on the specified ports/pins, or a virtual clock if no source other than a name is specified.

```
create_clock -period period_value [-name clock_name]
[-waveform> edge_list][source_objects]
```

Arguments

`-period period_value`

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-name clock_name`

Specifies the name of the clock constraint. You must specify either a clock name or a source.

`-waveform edge_list`

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

`source_objects`

Specifies the source of the clock constraint. The source can be ports, pins, or nets in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. You must specify either a source or a clock name.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Exceptions

- None

Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} -period 6 -waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock -period 7 -waveform {2 4} [get_ports {CK3}]
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

create_generated_clock

Creates an internally generated clock constraint on the ports/pins and defines its characteristics.

```
create_generated_clock [-name name] -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source
```

Arguments

`-name` ***name***

Specifies the name of the clock constraint.

`-source` ***reference_pin***

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by` ***divide_factor***

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` ***multiply_factor***

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} -divide_by 2 -source [get_ports {CLK}]
 U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock -divide_by 3 -multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

See Also

create_clock

Tcl Command Documentation Conventions

create_scenario

Creates a new timing scenario with the specified name. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
create_scenario name
```

Arguments

name

Specifies the name of the new timing scenario.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Description

A timing scenario is a set of timing constraints used with a design. Scenarios enable you to easily refine the set of timing constraints used for Timing-Driven Place-and-Route, so as to achieve timing closure more rapidly.

This command creates an empty timing scenario with the specified name and adds it to the list of scenarios.

Exceptions

- None

Example

```
create_scenario scenario_A
```

See Also

clone_scenario

Tcl Command Documentation Conventions

delete_scenario

Deletes the specified timing scenario.

```
delete_scenario name
```

Arguments

name

Specifies the name of the timing scenario to delete.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Description

This command deletes the specified timing scenario and all the constraints it contains.

Exceptions

- At least one timing scenario must always be available. If the current scenario is the only one that exists, you cannot delete it.
- Scenarios that are linked to the timing analysis or layout cannot be deleted.

Example

```
delete_scenario scenario_A
```

See Also

create_scenario

Tcl Command Documentation Conventions

Export

The syntax and arguments for the **export** command vary depending on which device you are designing. Click the appropriate command in the following list to see the syntax, arguments, and other information:

- export (IGLOO, Fusion, and ProASIC3)
- export (ProASIC^PLUS,ProASIC, Axcelerator, MX, eX, and SX/SX-A)
- export (Block support for IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S)

Export (IGLOO, Fusion, and ProASIC3)

Saves your design to a file in the specified file format. The required and optional arguments this command takes depends on which file format you specify.

```
export [-format value] [-feature value] [-secured_device value][-signature value][-pass_key
value][-aes_key value][-from_config_file value][-number_of_devices value][-
from_progfile_type value][-target_programmer value][-custom_security value][-
fpga_security_level value][-from_security_level value][-security_permanent value]{filename}
[-from_program_pages value][-from_content value] [-io_state value] [-efm_block_security
{location:X;security_level: value}] [-efm_content {location:X;source:value}] [-efm_block
{location:X;config_file:{value}}] [-efm_client {location:X;client: value;mem_file: value}}]
```

Arguments

`-format value`

Specifies the file format of the file to export. You can export one of the following types of files: edif, afm, dio, fus, log, sdf, adl, afl, cob, crt, dcf, design_script, loc, pin, session_script, stf, tcl, verilog, vhdl, crt, dcf, stp, pdc, stamp, gcf, prb, or sdc.

`-feature value`

Select the silicon feature(s) you want to program. Possible values for this option are all, setup_security, prog_from, prog_fpga, or the instance-specific program options available only for Fusion (listed below). Actel recommends that you specify your program parameters for each Embedded Flash Memory Block (EFMB) instance, from 0-3. The instance specific program options replace [-feature *value*].

`-secured_device value`

Specifies whether the device you are programming is secured. You can specify yes or no to enable or disable secured programming.

`-signature value`

Optional argument that identifies and tracks Actel designs and devices.

`-pass_key value`

Protects all the security settings for FPGA Array, FlashROM, and Embedded Flash Memory Block. The maximum length of this value is 32 characters. You must use hexadecimal characters for the pass key value.

`-aes_key value`

Decrypts FPGA Array and/or FlashROM and Embedded Flash Memory Block programming file content.

Max length is 32 HEX characters.

`-from_config_file value`

Specifies the location of the FlashROM configuration file.

`-number_of_devices value`

Specifies the number of devices you want to program. Applicable only when FlashROM has serialization regions.

`-from_progfile_type value`

Applicable only when FlashROM has serialization regions and STAPL file generation. Possible values:

Value	Description

Value	Description
single	Generates one programming file with all the generated incremental value(s) in the external source file
multiple	Generates one individual programming file for each generated incremental value(s) in the external source file

`-target_programmer` *value*

Applicable only when FlashROM has serialization regions and STAPL file generation. Possible values:

Value	Description
specific	Silicon Sculptor, BP Auto Programmer, or FlashPro
generic	Generic STAPL player

`-custom_security` *value*

Possible values:

Value	Description
yes	Custom security level
no	Standard security level

`-fpga_security_level` *value*

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FPGA Array cannot be written or verified without a Pass Key
write_protect	The security level is write protected. The FPGA Array cannot be written without a Pass Key, but it is open for verification (custom FPGA)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FPGA Array can be written and verified without a Pass Key

-from_security_level *value*

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FlashROM cannot be read, written or verified without a Pass Key
write_protect	The security level is write protected. The FlashROM cannot be written without a Pass Key, but it is open for reading and verification (custom FlashROM)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FlashROM can be written and verified without a Pass Key

`-security_permanent value`

Specifies whether the security settings for this file are permanent or not. Possible values:

Value	Description
yes	Permanently disable future modification of security settings for FPGA Array and FlashROM
no	Enable future modifications for FPGA Array and FlashROM

`-from_program_pages "value"`

Specifies FROM program pages in FlashPoint. If you use FlashROM content from an ADB file and do not specify a value, FlashPoint uses the same pages that were selected for programming in the previous FlashPoint session. Value may be a sequence of page numbers ("123") without a delimiter, or you can use any character or space as a delimiter, as in -from_program_pages "1 2 3".

You must specify pages for programming if you want FlashROM content from the UFC file.

`-from_content "value"`

Identifies the source file for the FlashROM content- a UFC or ADB file.

If this Tcl parameter is missing, FlashPoint tries to use the ADB as a source of FROM configuration and content data.

Values are shown in table below:

Value	Description
adb	(default)FROM content is taken from your ADB. Configurations from your UFC and ADB files are not compared.
ufc	FlashPoint uses FROM configuration and FROM content from the specified UFC file

`-io_state value`

The following table shows the acceptable values for this option. Values from both columns are acceptable. Value column lists def variables, and Description lists the GUI value.

Value	Description
Z	Tri-State
last_known_state	Last Known State
1	High
0	Low

`-efm_block_security{location:X;security_level: value}`

This option is available only for Fusion; X identifies an Embedded Flash Memory Block instance from 0-3.

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the Embedded Flash Memory Block cannot be read, written or verified without a Pass Key
write_protect	The security level is write protected. The Embedded Flash Memory Block cannot be written without a Pass Key, but it is open for reading (custom FB)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The Embedded Flash Memory Block can be written and read without a Pass Key

`-efm_content {location:X;source: value}`

This option is available only for Fusion; X identifies an Embedded Flash Memory Blockinstance from 0-3. Option identifies the source file for the Embedded Flash Memory Block content, either an EFC or ADB file.

If you wish to program the entire Embedded Flash Memory Block including all its clients that were programmed in previous sessions, and use ADB content for this client, this is the only parameter you must specify. If you wish to program the entire Embedded Flash Memory Block including all its clients and use the Embedded Flash Memory Block map file (EFC) you also have to specify the –efm_block parameter.

Possible values:

Value	Description
adb	(default) Embedded Flash Memory Block content is taken from your ADB
efc	FlashPoint uses the Embedded Flash Memory Block instance configuration and content from the EFC file specified in -efm_block_parameter

`-efm_block {location:X;source: value}`

This option is available only for Fusion; X identifies an Embedded Flash Memory Block (EFMB) instance from 0-3.

Config_file specifies the location of the EFMB instance configuration file (must be an EFC file with full pathname).

`-efm_client {location:X;client:value; mem_file: value}`

This option is available only for Fusion; X identifies an EFMB instance from 0-3.

You must specify the client name and its memory content file for each client of EFMB you wish to program.

Mem_file specifies the file with the memory content for the client. If a mem_file path is specified, the memory content from this file will overwrite the client content in ADB or EFC (as defined by the -efm_content argument). If the client memory file is not specified, the client memory content from the ADB or EFC file is used instead (as defined by the -efm_content argument).

`{filename}`

Specifies the path and name of the file you are exporting.

Supported Families

IGLOO, Fusion, and ProASIC3

Notes

- None

Exceptions

- None

Example

```
export -format "bts_stp"
-feature "all"
-secured_device “no”
-signature "123"
-pass_key "FB318707864EC889AE2ED8904B8EB30D"
-custom_security "no"
-fpga_security_level "write_verify_protect"
-from_security_level "write_verify_protect"
-from_config_file {.\g3_test\from.ufc}
-number_of_devices "1"
-from_progfile_type "single"
-target_programmer "specific" \
 {.\flp4.stp}
```

Fusion example 1:

Export soc.pdb file that includes programming data for three clients of EFM block 0. EFM block configuration file ./fus_new/nvm_simple/nvm_simple.efc and clients memory files are used for generating the programming file. Clients specified as TCL parameters must be included in EFC file.

```
export -format "pdb "
-efm_content {location:0; source:efc} \
-efm_block {location:0; config_file:{./fus_new/nvm_simple/nvm_simple.efc}} \
-efm_client {location:0; client:cfiData;
```

```
mem_file:{./fus_new/nvm_exmp/input_memfiles/ram1_block_0_ram1_R0C0.mem}} \
-efm_client {location:0; client:dataStorage;
mem_file:{./fus_new/nvm_exmp/input_memfiles/datast2_asb1_smtr_ram.hex}} \
-efm_client {location:0; client:init1;
mem_file:{./fus_new/nvm_exmp/input_memfiles/datast1_asb1_acm_rtc_ram.hex}} \
 {./soc} Fusion example 2:
```

Export soc.stp and soc.pdb files that include programming data for EFM block 0. Information regarding block configuration, which clients to program, and their memory content is taken from ADB file.

```
export -format "pdb bts_stp"
-efm_content {location:0; source:adb} \
 {./soc}
```

Fusion example 3:

Export soc.stp and soc.pdb files that include programming data for client cfiData of EFM block 0. Other clients of block 0 are not selected to be programmed. ADB file is a source for block configuration and content; EFC is ignored.

```
export -format "pdb"
-efm_content {location:0; source:adb} \
-efm_block {location:0; config_file:{./fus_new/nvm_simple/nvm_simple.efc}} \
-efm_client {location:0; client:cfiData;} \
 {./soc}
```

See Also

export (ProASIC[PLUS], Axcelerator, ProASIC, MX, eX, and SX/SX-A)

Exporting files

Importing files

Tcl documentation conventions

Export (ProASIC[PLUS],ProASIC, Axcelerator, MX, eX, and SX/SX-A)

Saves your design to a file in the specified file format. The required and optional arguments this command takes depends on which file format you specify.

```
export -format file_type{filename}
export -format edif -edif_flavor value {filename}
export -format file_type [-signature value] {filename}
export -format log [-diagnostic] (filename}
export -format sdf [-prelayout] {filename} -axprg_set_algo {programming algorithm}
```

Arguments

`-format file_type`

Specifies the file format of the file to export. You can export one of the following types of files: edif, afm, dio, fus, log, sdf, adl, afl, cob, crt, dcf, design_script, loc, pin, session_script, stf, tcl, verilog, vhdl, crt, dcf, stp, pdc, stamp, gcf, prb, or sdc.

`-edif_flavor (value)`

The acceptable values for this argument are generic, viewlogic, mgc, orcad, or workview.

`-signature value`

Optional for afm, dio, and fus file types.

`-diagnostic`

Optional for Log files. Specifies that you want detailed messages exported to a Log file.

`-prelayout`

Optional for .sdf files.

`-axprg_set_algo {programming algorithm}`

Sets the programming algorithm for your RTAX250S, RTAX1000S, or RTAX2000S device. Options are described in the table below.

Value	Description
OPA	Original programming algorithm; generates the OPA AFM file for your device.
UMA	UMC modified algorithm; generates the UMC AFM file for your device

`{filename}`

Specifies the path and name of the file you are exporting.

Notes

- When exporting a Tcl script, remove the design_script and session_script values as follows:

```
export -format tcl -scope session
export -format tcl -scope design
```

"Session" refers to the Tcl commands of the entire Designer session; a session in Designer starts when you open the application and ends when you close it. "Design" refers to the Tcl commands for the current design only.

Supported Families

ProASIC[PLUS],ProASIC, Axcelerator, MX, eX, and SX/SX-A

Notes

- None

Exceptions

- None

See Also

export (IGLOO, Fusion,and ProASIC3 families)

Exporting files

Importing files

Tcl documentation conventions

Export (Designer Block support for IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX families)

Exports (publishes) the Designer Block files to a specified directory, includes any added comments.

```
export -format "block"
-export_directory {value} \
-export_name "blockname" \
-placement "value"\
-routing "value"\
-comment "value" \
-export_language "value"\
-region "value"
```

Arguments

`-export_directory {value}`

Specifies the directory name for the exported *.v, *.vhd, *.cxf and *.cdb files. Value is the path and name of the directory

`-export_name "blockname"`

Specifies the prefix of the exported *.v, *.vhd, *.cxf, and *.cdb files, where blockname is the name of the prefix.

`-placement "value"`

Exports placement information. Possible values:

Value	Description
yes	Exports the placement information. Specify "yes" only if the placer state is valid and -placement is specified as "yes."
no	Do not export the placement information.

`-routing "value"`

Exports placement information. Possible values:

Value	Description
yes	Exports routing information. Specify "yes" only if the routing state is valid and -placement is specified as "yes."
no	Do not export the routing information.

`-comment "value"`

Adds comments to document the block.

`-export_language "value"`

Specifies the export format of the CXF file for Libero IDE. Possible values:

Value	Description
VERILOG	CXF file is Verilog.
VHDL	CXF file is VHDL.

`-region "value"`

Option to publish all the user regions and make them available when you instantiate the block. Possible values:

Value	Description
YES	Publishes all the user regions, makes them available when you instantiate your block.
NO	Disables region publishing

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX

Exceptions

- None

Example

```
export -format "block" -export_directory {.} -export_name "test_core" -placement "yes" -
routing "yes" -comment "toto" -export_language "VERILOG"
```

See Also

Exporting files

Importing files

Tcl documentation conventions

get_cells

Returns an object representing the cells (instances) that match those specified in the pattern argument.

```
get_cells pattern
```

Arguments

`pattern`

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where "*" is a wildcard that represents any character string.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_max delay, set_multicycle_path, and set_false_path.

Exceptions

- None

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path –through [get_cells {Rblock/muxA}]
```

See Also

get_clocks

get_nets

get_pins

get_ports

Tcl Command Documentation Conventions

get_clocks

Returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to use to match the clocks set in SmartTime or Timer.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Description

- If this command is used as a –from argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Exceptions

- None

Example

```
set_max_delay -from [get_ports data1] -to \
[get_clocks clk1]
```

See Also

create_clock

create_generated_clock

Tcl Command Documentation Conventions

get_current_scenario

Returns the name of the current timing scenario.

```
get_current_scenario
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Exceptions

None

Examples

```
get_current_scenario
```

See Also

set_current_scenario

Tcl documentation conventions

get_defvar

Provides access to the internal variables within Designer and returns its value. This command also prints the value of the Designer variable on the Log window.

```
get_defvar variable
```

Arguments

variable

The Designer internal variable.

Supported Families

All

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

Retrieves the full qualified path of the design file. 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.

```
get_design_filename
```

Arguments

None

Supported Families

All

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

Retrieves some basic details of your design. The result value of the command will be a string value.

```
get_design_info value
```

Arguments

value

Must be one of the valid string values summarized in the table below:

Value	Description
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	Fully 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. 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.

Supported Family

All

Exceptions

- Returns an error if a design is not loaded.
- Returns an error if more than one argument is passed.
- Returns 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

get_nets

Returns an object representing the nets that match those specified in the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where "*" is a wildcard that represents any character string.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as `-through` arguments in the set false path, set minimum delay, set maximum delay, and set multicycle path constraints.

Exceptions

- None

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path -through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -period 2.5 [get_nets {cknet}]
```

See Also

create_clock

create_generated_clock

set_false_path

set_min_delay

set_max_delay

set_multicycle_path

Tcl documentation conventions

get_out_of_date_files

Audits all files returns a list of filenames that are out of date; each filename is separated by a space. The command returns a string of file names that are out of date separated by a space

i.e. file1 file2 ...

It returns empty string if all files are current.

This command ignores the Audit settings in your ADB file.

```
get_out_of_date_files
```

Arguments

None

Supported Family

All

Exceptions

- None

Example

The following code returns a list of filenames that are out of date.

```
get_out_of_date_files
```

See Also

are_all_source_files_curent

is_source_file_current

get_pins

Returns an object representing the pin(s) that match those specified in the pattern argument.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins to return. For example, "get_pins clock_gen*" returns all pins starting with the characters "clock_gen", where "*" is a wildcard that represents any character string.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

- None

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

See Also

create_clock

create_generated_clock

set_clock_latency

set_false_path

set_min_delay

set_max_delay

set_multicycle_path

Tcl documentation conventions

get_ports

Returns an object representing the port(s) that match those specified in the pattern argument.

```
get_ports pattern
```

Argument

pattern

Specifies the pattern to match the ports.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

- None

Example

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

See Also

create_clock

set_clock_latency

set_input_delay

import_aux

Imports the specified auxiliary file into the design. Equivalent to executing the Import Auxiliary Files command from the File menu.

```
import_aux
-format file_type
 -top_level_name top_level_name
-glitch_filter threshold
[-merge_timing] value
filename
```

Arguments

`-format file_type`

Specifies the file format of the file to import. You can import one of the following types of files: pdc, sdc, pin, dcf, saif, vcd, or crt.

`-top_level_name top_level_name`

Specifies the instance name of your design in the simulation testbench when you import a VCD or SAIF file.

When importing a VCD file, the automatic *top_level_name* detection is available. If the -top_level_name option is not specified, SmartPower will try to automatically detect the top level name.

When importing a SAIF file, the automatic *top_level_name* detection is not available and -top_level_name is a required argument.

To identify the top_level_name for SAIF and VCD files manually, refer to Importing a VCD file and Importing a SAIF file.

`-glitch_filter threshold`

Specifies the glitch option/ filtering threshold. To disable glitch filtering, remove `-glitch_filter` from the command line. The following table shows the acceptable values for the this option:

Value	Description
auto	Enables automatic glitch filtering.
a positive integer	Enables the glitch filtering option with the specified integer as threshold in picoseconds (ps).

`-merge_timing value`

Specifies whether to preserve all existing timing constraints when you import an SDC file. Same as selecting or unselecting the "Keep existing timing constraints" check box in the Import Files dialog box. The following table shows the acceptable values for the this option:

Value	Description
yes/true	Designer merges the timing constraints from the imported SDC file with the existing constraints saved in the constraint database.
no/false	The existing timing constraints are replaced by the constraints in the newly imported SDC file.

`filename`

Specifies the name of the auxiliary file to import.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, MX, eX, SX/SX-A

Description

- Auxiliary files are not audited and are handled as one-time data-entry or data-change events, similar to entering data using one of the interactive editors (for example, PinEditor or Timer).
- If you import the SDC file as an auxiliary file, you do not have to re-compile your design. However, auditing is disabled when you import auxiliary files, and Designer cannot detect the changes to your SDC file(s) if you import them as auxiliary files.

Exceptions

- None

Examples

```
import_aux -format sdc file.sdc
import_aux -format pdc file.pdc
import_aux -format vcd file.vcd // automatic detection of top level name
import_aux -format vcd -glitch_filter 10 // filter out glitches that are 10 ps or less
import_aux -format saif -top_level_name "top" file.saif
```

See Also

import_source

Importing auxiliary files

Importing source files

Importing files

Tcl documentation conventions

import_source

Imports the specified source file into the design. Equivalent to executing the Import Source File command from the File menu.

All source files must be specified on one command line.

```
import_source [-merge_timing value][-merge_physical value][-merge_all value][-format
file_type][-abort_on_error value][-top_entity][-edif -edif_flavor value]filename
```

Arguments

`-merge_timing value`

Specifies whether to preserve all existing timing constraints when you import an SDC file. Same as selecting or unselecting the "Keep existing timing constraints" check box in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description
yes	Designer merges the timing constraints from the imported SDC file with the existing constraints saved in the constraint database. If there is a conflict, the new constraint has priority over the existing constraint.
no	All existing timing constraints are replaced by the constraints in the newly imported SDC file.

`-merge_physical value`

Specifies whether to preserve all existing physical constraints when you import a GCF or PDC file. Same as selecting or unselecting the "Keep existing physical constraints" check box in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description
yes	Designer preserves all existing physical constraints that you have entered either using one of the MVN tools (ChipPlanner, PinEditor, or the I/O Attribute Editor) or a previous GCF or PDC file. The software resolves any conflicts between new and existing physical constraints and displays the appropriate message.
no	All existing physical constraints are replaced by the constraints in the newly imported GCF or PDC file.

`-merge_all value`

Specifies whether to preserve all existing physical and timing constraints when you import an SDC and/or a PDC file. Same as selecting or unselecting the "Keep existing physical constraints" and "Keep existing timing constraints" check boxes in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description

Value	Description
yes	Designer preserves all existing physical constraints that you have entered either using one of the MVN tools (ChipPlanner, PinEditor, or the I/O Attribute Editor) or a previous GCF or PDC file. The software resolves any conflicts between new and existing physical constraints and displays an appropriate message. Any existing timing constraints from your ADB are merged with the new information from your imported files. New constraints override any existing timing constraints whenever there is a conflict
no	All the physical constraints in the newly imported GCF or PDC files are used. All pre-existing physical constraints are lost. Existing timing constraints from the ADB are replaced by the new timing constraints from your imported file.

`-format` *`file_type`*

Specifies the file format of the file to import. You can import one of the following types of files: adl, edif, verilog, vhdl, gcf, pdc, sdc, or crt.

Note: Refer to Importing source file to know the formats supported for each family.

`-abort_on_error` *`value`*

Aborts a PDC file if it encounters an error during import. Possible values are

Value	Description
yes	Designer aborts on error.
no	Designer ignores the error and continues.

`-top_entity`

Specifies the top entity to a VHDL file.

`-edif edif_flavor` *`value`*

Specifies the type of netlist. It can be edif, viewlogic, or mgc.

`filename`

Specifies the name of the source file to import.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], Axcelerator, ProASIC, MX, eX, SX/SX-A

Exceptions

Your script -merge options vary according to family as shown below:

- The -merge_timing, -merge_physical, and -merge_all arguments are available for IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator families.
- For source SDC (no PDC/GCF) in eX and SX-A:

```
import_source -merge_timing yes/no ...
import_source -merge yes/no  ...
import_source -merge_all yes/no  ...
```

The eX and SX-A families do not support merging of physical constraints. For these families, in import_source, the -merge_all and -merge options are mapped just to -merge_timing.

- For constraints in ProASIC^{PLUS}:

```
import_source -merge_physical yes/no ...
import_source -merge_all yes/no ...
import_source -merge yes/no ...
```

ProASIC^{PLUS} and ProASIC support GCF, but the -merge option in the import_source only affects GCF in terms of the physical constraints.

- ProASIC does not support PreCompile import of the SDC files. The -merge_timing option has no effect on this import_source for ProASIC. The -merge_all and -merge options map to -merge_physical for ProASIC in import_source.
- For IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, and Axcelerator:

```
import_source -merge_physical yes/no -merge_timing yes/no ...
import_source -merge_all yes/no  ...
import_source -merge yes/no  ...
```

The -merge_all and -merge options map to both -merge_physical and -merge_timing options for these families.

Examples

Consider the following sample scripts:

```
import_source  \
 -merge_physical "no" \
 -merge_timing "yes"
  -format "EDIF" -edif_flavor "GENERIC" \
 {.\designs\mydesign.edn} \
 -format "sdc" \
 {.\designs\mydesign.sdc} \
 -format "pdc" -abort_on_error "no" \
 {.\designs\mydesign.pdc}

import_source \
 -merge_physical "no" \
 -format "verilog" \
```

```
 {mydesign.v}

import_source \
  -merge_physical "no" \
  -merge_timing "no" \
  -format "vhdl" -top_entity "aclass" \
   {C:/mynetlist.vhd}

import_source \
  -merge_physical "no" \
  -merge_timing "no" \
  -format "adl" {mydesign.adl}
```

See Also

import_aux

Importing auxiliary files

Importing source files

Importing files

Tcl documentation conventions

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.

```
is_design_loaded
```

Arguments

None

Supported Family

All

Description

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

Returns a Boolean value (0 for false, 1 for true) indicating if a design has been modified in the Designer software. True is returned if a design has been modified.

```
is_design_modified
```

Arguments

None

Supported Family

All

Description

Some Tcl commands are valid only if a design has been modified in Designer. Use the is_design_modified command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

- Returns an error if arguments are passed.

Example

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

```
set bDesignModified [ is_design_modified ]
if { $bDesignModified == 0 } {
  puts "Design has not been modified."
}
```

See Also

get_design_filename
get_design_info
is_design_loaded
is_design_state_complete

is_design_state_complete

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.

```
is_design_state_complete value
```

Arguments

value

Must be one of the valid string values summarized in the table below:

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

Supported Family

All

Description

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

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 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

is_source_file_current

Audits the source file and determines whether or not the file is out of date / imported into the workspace. Returns '0' if file_name is out of date or has not been imported into the workspace, and returns '1' if file_name is current.

This command ignores the Audit settings in your ADB file.

```
is_source_file_current(filename)
```

Arguments

filename is the path to the source file

Supported Families

All

Exceptions

- None

Example

The following code determines whether or not the file has been imported into the workspace.

```
is_source_file_current (./hdl/adder.vhd)
```

See Also

are_all_source_files_curent
get_out_of_date_files

Layout - IGLOO, Fusion, and ProASIC3

This command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[-power_driven value]
[-run_placer value]
[-place_incremental value]
[-run_router value]
[-route_incremental value]
```

Arguments

`-timing_driven|-standard`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`-power_driven value`

The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout. This is the default.
on	Enables power-driven layout

`-place_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "fixed" and continues to place the remaining ones

`-route_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Skips incremental mode, discards previous information. This is the default.
on	Invokes incremental routing and sets the previous routing information as the initial starting point

`-run_placer` *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placement. This is the default.
off	Skips placement

`-run_router` *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes routing if placement is successful. This is the default.
off	Skips routing

Layout - Advanced Options for IGLOO, Fusion, and ProASIC3

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

```
[-placer_high_effort value]
[-seq_opt  value]
[-mindel_repair value]
[-placer_seed value]
[-show_placer_seed]
```

Arguments

`-placer_high_effort` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Disables physical synthesis of combinational logic. This is the default.
on	Enables physical synthesis of combinational logic

`-seq_opt` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Disables physical synthesis of sequential logic. This is the default.
on	Enables physical synthesis of sequential logic in high-effort mode

`-mindel_repair` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Does not run minimum delay violations repair. This is the default.
on	Enables repair of minimum delay violations during route

`-placer_seed` *value*

An integer value that you can set to change the initial random seed number for the placement.

`-show_placer_seed` *value*

Causes Layout to display the initial random seed number used for the placement.

Exceptions

- None

Example

```
layout
layout -place_incremental FIX -route_incremental ON
layout -placer_high_effort ON
layout -run_placer OFF -route_incremental ON -mindel_repair ON
layout -timing_driven -power_driven ON
layout -placer_seed 120
```

See Also

IGLOO, Fusion, and ProASIC3 Layout options

IGLOO, Fusion, and ProASIC3 Advanced Layout options

Layout - ProASIC and ProASIC^PLUS

This command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[-place_incremental value]
[-route_incremental value]
[-run_placer value]
[-run_router value]
```

Arguments

`-timing_driven|-standard`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`-place_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "FIXED" and continues to place the remaining ones

`-route_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Skips incremental mode, discards previous information. This is the default.
on	Invokes incremental mode, and sets the previous routing information as the initial starting point

-run_placer *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placement. This is the default.
off	Skips placement

-run_router|-router|-route *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes routing if placement is successful. This is the default.
off	Skips routing

Layout - Advanced Layout Options for ProASIC and ProASIC[PLUS]

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

```
[-timing_weight value]
[-placer_seed value]
[-show_placer_seed]
```

Arguments

-timing_weight *value*

The timing weight for the placement. Values range from 1 to 4 in whole integers. The lower numbers give less weight to timing in the placement. Timing weight 4 is the same as timing driven placement using the option –timing_driven.

-placer_seed *value*

An integer value that you can set to change the initial random seed number for the placement.

-show_placer_seed

Causes Layout to display the initial random seed number used for the placement.

Exceptions

- None

Example

```
layout
layout -standard
layout -timing_weight 3
layout -place_incremental FIX -route_incremental ON
layout -placer_seed 120
```

See Also

ProASICPLUS and ProASIC Layout Options

ProASICPLUS, ProASIC Advanced Layout Options

Layout - Axcelerator

This command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[-power_driven value]
[-place_incremental value]
[-effort_level value]
[-route_incremental value]
[-run_placer value]
[-run_router value]
```

Arguments

`-timing_driven|-standard`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`-power_driven value`

The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout. This is the default.
on	Enables power-driven layout

`-place_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "fixed" and continues to place the remaining ones

`-effort_level value`

The effort level for the placement. Values range from 1 to 5 in whole integers. The lower numbers tend to give a result more quickly, but higher levels yield better performance in timing and routability.

`-route_incremental value`

The following table shows the acceptable values for this argument:

Value	Description
off	Skips incremental mode, discards previous information. This is the default.
on	Invokes incremental mode, and sets the previous routing information as the initial starting point

`-run_placer value`

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placement. This is the default.
off	Skips placement

`-run_router value`

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes routing if placement is successful. This is the default.
off	Skips routing

Layout - Advanced Options for Axcelerator

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

```
[-mindel_repair value]
[-placer_seed value]
[-show_placer_seed]
```

Arguments

`-mindel_repair value`

The following table shows the acceptable values for this argument:

Value	Description
off	Skips minimum delay violations repair. This is the default.
on	Invokes minimum delay violations repair

-placer_seed *value*

An integer value that you can set to change the initial random seed number for the placement.

-show_placer_seed

Causes Layout to display the initial random seed number used for the placement.

Exceptions

N/A

Example

```
layout
layout -timing_driven -effort_level 4
layout -place_incremental FIX -route_incremental ON
layout -placer_seed 120
layout -mindel_repair ON
```

See Also

Axcelerator Layout options

Axcelerator Advanced Layout options

Layout - SX-A, eX, and SX

This command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[-place_incremental value]
```

Arguments

-timing_driven|-standard

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

-place_incremental *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "FIXED" and continues to place the remaining ones

Layout - Advanced Options for SX-A, eX, and SX

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

```
[-extended_run value]
[-effort_level value]
[-timing_weight value]
[-placer_seed value]
[-show_placer_seed value]
```

Arguments

`-extended_run value`

The following table shows the acceptable values for this argument:

Value	Description
off	Skips extended run mode. This is the default.
on	Invokes extended run mode

`-effort_level value`

Sets the effort level; number may range from 25 to 500.

`-timing_weight value`

Sets the timing weight value; number may range from 10 to 150.

`-placer_seed value`

An integer value that you can set to change the initial random seed number for the placement.

`-show_placer_seed`

Causes Layout to display the initial random seed number used for the placement.

Exceptions

- None

Example

```
layout [-timing_driven] [-incremental OFF] [-extended_run OFF] [-effort_level 50] [-
timing_weight 100]
```

See Also

eX, SX, and SX-A Layout options

eX, SX, and SX-A Advanced Layout options

Layout - MX, DX, ACT

This command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[-place_incremental value]
```

Arguments

`-timing_driven|-standard`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`-place_incremental` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "fixed" and continues to place the remaining ones

Layout - Advanced Options for MX, DX, ACT

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

```
[-extended_run value]
[-placer_seed value]
[-show_placer_seed]
```

Arguments

`-extended_run` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Skips extended run mode. This is the default.
on	Invokes extended run mode

`-placer_seed` *value*

An integer value that you can set to change the initial random seed number for the placement.

`-show_placer_seed`

Causes Layout to display the initial random seed number used for the placement.

Exceptions

- None

Example

The following code runs layout and specifies the timing-driven option:

```
layout -timing_driven
```

See Also

MX, DX, and ACT Layout options

MX, DX, and ACT Advanced Layout options

list_clocks

Returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_clocks]
```

See Also

create_clock

remove_clock

Tcl documentation conventions

list_clock_latencies

Returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency

remove_clock_latency

Tcl documentation conventions

list_disable_timings

Returns the list of disable timing constraints for the current scenario.

```
list_disable_timings
```

Arguments

- None

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Exceptions

- None

Example

```
list_disable_timings
```

list_false_paths

Returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_false_paths]
```

See Also

set_false_path

remove_false_path

Tcl documentation conventions

list_generated_clocks

Returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_generated_clocks]
```

See Also

create_generated_clock

remove_generated_clock

Tcl documentation conventions

list_input_delays

Returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_input_delays]
```

See Also

set_input_delay

remove_input_delay

Tcl documentation conventions

list_max_delays

Returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay

remove_max_delay

Tcl documentation conventions

list_min_delays

Returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay

remove_min_delay

Tcl documentation conventions

list_multicycle_paths

Returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path

remove_multicycle_path

Tcl documentation conventions

list_objects

Returns a list of names of the objects in the specified list.

```
list_scenarios <object_list>
```

Arguments

None

Supported Families

All

Exceptions

None

Examples

```
list_objects <object_list>
```

See Also

Tcl documentation conventions

list_output_delays

Returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay

remove_output_delay

Tcl documentation conventions

list_scenarios

Returns a list of names of all of the available timing scenarios.

```
list_scenarios
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Exceptions

None

Examples

```
list_scenarios
```

See Also

get_current_scenario

Tcl documentation conventions

LOGFILE

The LOGFILE command is not a Tcl script. It runs in conjunction with your Tcl script and enables you to specify a filename to which Designer will record/save a copy of the log messages generated in batch-mode (SCRIPT:...).

It is useful if you want to view the log after you run scripts in batch-mode on Windows.

For example, to run the script 'myscript.tcl' for Designer and save the log messages to a LOGFILE named 'mylog.txt', use the command:

```
designer.exe SCRIPT:myscript.tcl LOGFILE:mylog.txt
```

See Also

Introduction to Tcl scripting

new_design

Creates a new design. You need all three arguments for this command. This command will set up the Designer software for importing design source files

```
new_design -name design_name -family family_name -path  pathname-block value
```

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.

`block value`

Enables or disables Block mode.The following table shows the acceptable values for this option:

Value	Description
on	Enables Block mode
off	Disables Block mode

Supported Families

All

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

close_design

open_design

save_design

set_design

open_design

Opens an existing design into the Designer software.

```
open_design file_name
```

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

Arguments

file_name

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

Supported Families

All

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

close_design

new_design

save_design

pin_assign

Use to either assign the named pin to the specified port or assign attributes to the specified port. This command has two syntax formats. The one you use depends on what you are trying to do. The first syntax format assigns the named pin to the specified port and supports all families. The second one assigns attributes to the specified port but supports only the ProASIC3, Axcelerator, SX-A, RTSX-S, and eX families.

```
pin_assign [-nofix] -port portname -pin pin_number
pin_assign -port portname [-iostd value][-iothresh value][-outload value][-slew value][-
res_pull value]
```

Arguments

-nofix

Unlocks the pin assignment (by default, assignments are locked).

-port *portname*

Specifies the name of the port to which the pin is assigned.

-pin *pin_number*

Specifies the alphanumeric number of the pin to assign.

-iostd *value*

Sets the I/O standard for this pin. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the supported devices:

I/O Standards table

Use the I/O Standards table to see which I/O standards can be applied to each family:

I/O Standard	IGLOO	Fusion	ProASIC3	Axcelerator	RTSX-S	SX-A
CMOS					X	
CUSTOM					X	X
GTLP25	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
GTLP33	IGLOO e only	X	ProASIC3E and ProASIC3L only			
GTL33	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
GTL25	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
HSTL1	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
HSTLII	IGLOO e only	X	ProASIC3E and ProASIC3L only			
LVCMOS33	X	X	X			
LVCMOS25	IGLOO e only	X	X	X		
LVCMOS25_50	X	X	X			
LVCMOS18	X	X	X	X		
LVCMOS15	X	X	X	X		
LVCMOS12	X		ProASIC3L only			
LVTTL	X	X	X	X	X	X
TTL	X	X	X	X	X	X
PCI	X	X	X	X	X	X
PCIX	X	X	X	X		

I/O Standard	IGLOO	Fusion	ProASIC3	Axcelerator	RTSX-S	SX-A
SSTL2I	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
SSTL2II	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
SSTL3I	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		
SSTL3II	IGLOO e only	X	ProASIC3E and ProASIC3L only	X		

See Also

I/O standard

Note: The LVDS and LVPECL I/O standards cannot be set through a script.

`-iothresh value`

Sets the compatible threshold level for inputs and outputs. The default I/O threshold is based upon the I/O standard. You can set the I/O Threshold independently of the I/O specification in the PinEditor tool by selecting **CUSTOM** in the I/O Standard cell. The following table shows the acceptable values for the supported devices (SX-A, RTSX-S, and eX):

Value	Description
CMOS	RTSX-S devices only. An advanced integrated circuit (IC) manufacturing process technology for logic and memory, characterized by high integration, low cost, low power, and high performance. CMOS logic uses a combination of p-type and n-type metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits found in computers, telecommunications, and signal processing equipment.
LVTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3V applications. It uses an LVTTL input buffer and a push-pull output buffer.
PCI	A computer bus for attaching peripheral devices to a computer motherboard in a local bus. This standard supports both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding IGLOO, Fusion, and ProASIC3 families.

Note: The -iothresh attribute is also referred to as "Loading" in some families.

`-slew value`

Sets the output slew rate. Slew control affects only the falling edges. Rising edges are not affected. This attribute is only available for LVTTL, PCI, and PCI outputs. For LVTTL, it can either be high or low. For PCI and PCIX, it

can only be set to high. The following table shows the acceptable values for the supported devices (IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A):

Value	Description
high	Sets the I/O slew to high
low	Sets the I/O slew to low

`-res_pull` *value*

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. The following table shows the acceptable values for the supported devices (IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A):

Value	Description
up	Includes a weak resistor for pull-up of the input buffer
down	Includes a weak resistor for pull-down of the input buffer
none	Does not include a weak resistor

`-out_load` *value*

Indicates the output-capacitance value based on the I/O standard selected. This option is not available in SmartGen. This attribute determines what Timer will use as the loading on the output pin and applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). The default is 35*pF*. This attribute is available only for the following devices: Fusion, ProASIC3, Axcelerator, SX-A, RTSX-S, and eX.

Supported Families

The first syntax statement for pin_assign supports all families. The second one supports only IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A.

Exceptions

- None

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_assign -port usw0 -pin A2
pin_commit

pin_assign -port usw0 -iostd LVTTL -slew low -res_pull down
pin_commit
```

Note:

- To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket as shown in the following examples.

- The following example shows a port name enclosed with curly braces:
- The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

See Also

pin_commit

pin_fix

pin_unassign

Tcl documentation conventions

pin_commit

Saves the pin assignments to the design (.adb) file.

```
pin_commit
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

- None

Supported Families

All families

Exceptions

- None

Examples

To save pin assignments in your design, you must add the pin_commit command to the end of the script:

```
pin_commit
```

See Also

pin_fix

pin_unfix

pin_assign

pin_unassign

Tcl documentation conventions

pin_fix

Locks the pin assignment for the specified port, so the pins cannot be moved during place-and-route.

```
pin_fix -port portname
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

`-port portname`

Specifies the name of the port to which the pin must be locked at its assigned location.

Note: You can assign only one pin to a port

Supported Families

All families

Description

Fixed pins are locked pins. You cannot move locked pins during place-and-route.

Exceptions

- None

Examples

You must use `pin_commit` after the pin_fix command to save the changes to your design:

```
pin_fix -port clk
pin_commit
```

See Also

pin_commit

pin_unfix

pin_assign

pin_unassign

Tcl documentation conventions

pin_fix_all

Locks all the assigned pins on the device so they cannot be moved during place-and-route.

```
pin_fix_all
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

- None

Supported Families

All families

Description

Fixed pins are locked pins. This command locks all the pins in your design. You cannot move locked pins during place-and-route.

Exceptions

- None

Example

You must use `pin_commit` after the pin_fix_all command to save the changes to your design:

```
pin_fix_all
pin_commit
```

See Also

pin_commit

pin_fix

pin_unfix

pin_assign

pin_unassign

Tcl documentation conventions

pin_unassign

Unassigns the pin from the specified port. The unassigned pin location is then available for other ports. (Only one pin can be assigned to a port.)

```
pin_unassign -port portname
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

`-port portname`

Specifies the name of the port for which the pin must be unassigned.

Supported Families

All

Exceptions

- None

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_unassign -port "clk"
pin_commit
```

See Also

pin_commit

pin_fix

pin_fix_all

pin_unfix

pin_assign

pin_unassign

Tcl documentation conventions

pin_unassign_all

Unassigns all the pins from all the ports so that all pin locations are available for assignment.

```
pin_unassign_all
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

- None

Supported Families

All families

Exceptions

- None

Examples

You must use `pin_commit` after the pin_assign_all command to save the changes to your design:

```
pin_unassign_all
pin_commit
```

See Also

pin_commit

pin_fix

pin_unfix

pin_assign

pin_unassign

Tcl documentation conventions

pin_unfix

Unlocks the pins assigned to the specified port, so the pins can be moved during place-and-route.

```
pin_unfix -port portname
```

Note: You can use this command for designs created with PinEditor in MVN and PinEditor (non-MVN).

Arguments

`-port portname`

Specifies the name of the port containing pins to unlock.

Supported Families

All families

Exceptions

None

Examples

You must use `pin_commit command` after the pin_unfix command to save the changes to your design:

```
pin_unfix -port rst
pin_commit
```

See Also

pin_commit

pin_fix

pin_assign

pin_unassign

Tcl documentation conventions

remove_clock

Removes the specified clock constraint from the current timing scenario.

```
remove_clock {-name clock_name| -id constraint_ID
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Removes the specified clock constraint from the current scenario. If the specified name does not match a clock constraint in the current scenario, or if the specified ID does not refer to a clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

- You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock constraint named "my_user_clock":

```
remove_clock -name my_user_clock
```

The following example removes the clock constraint using its ID:

```
set clockId [create_clock -name my_user_clock -period 2]
remove_clock -id $clockId
```

See Also

create_clock

Tcl Command Documentation Conventions

remove_clock_latency

Removes a clock source latency from the specified clock and from all edges of the clock.

```
remove_clock_latency {-source clock_name_or_source |-id constraint_ID}
```

Arguments

`-source clock_name_or_source`

Specifies either the clock name or source name of the clock constraint from which to remove the clock source latency. You must specify either a clock or source name or its constraint ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either a clock or source name or its constraint ID.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Removes a clock source latency from the specified clock in the current scenario. If the specified source does not match a clock with a latency constraint in the current scenario, or if the specified ID does not refer to a clock with a latency constraint, this command fails.

Do not specify both the source and the ID.

Exceptions

- You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock source latency from the specified clock.

```
remove_clock_latency -source my_clock
```

See Also

set_clock_latency

Tcl Command Documentation Conventions

remove_disable_timing

Removes a disable timing constraint by specifying its arguments, or its ID. If the arguments do not match a disable timing constraint, or if the ID does not refer to a disable timing constraint, the command fails.

```
remove_disable_timing -from value -to value name -id name
```

Arguments

`-from from_port`

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to to_port`

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

`-id name`

Specifies the constraint name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Exceptions

- None

Example

```
remove_disable_timing -from port1 -to port2 -id new_constraint
```

remove_false_path

Removes a false path from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_false_path [-from from_list] [-to to_list] [-through through_list] [-id constraint_ID]
remove_false_path -id constraint_ID
```

Arguments

`-from` ***`from_list`***

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` ***`through_list`***

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to` ***`to_list`***

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` ***`constraint_ID`***

Specifies the ID of the false path constraint to remove from the current scenario. You must specify either the exact false path to remove or the constraint ID that refers to the false path constraint to remove.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

Removes a false path from the specified clock in the current scenario. If the arguments do not match a false path constraint in the current scenario, or if the specified ID does not refer to a false path constraint, this command fails.

Do not specify both the false path arguments and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an Accessor command such as get_pins or get_ports.

Examples

The following example specifies all false paths to remove:

```
remove_false_path -through U0/U1:Y
```

The following example removes the false path constraint using its id:

```
set fpId [set_false_path -from [get_clocks c*] -through {topx/reg/*} -to [get_ports
out15] ]
remove_false_path -id $fpId
```

See Also

set_false_path

Tcl Command Documentation Conventions

remove_generated_clock

Removes the specified generated clock constraint from the current scenario.

```
remove_generated_clock {-name clock_name | -id constraint_ID }
```

Arguments

`-name clock_name`

Specifies the name of the generated clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the generated clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Removes the specified generated clock constraint from the current scenario. If the specified name does not match a generated clock constraint in the current scenario, or if the specified ID does not refer to a generated clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

- You cannot use wildcards when specifying a generated clock name.

Examples

The following example removes the generated clock constraint named "my_user_clock":

```
remove_generated_clock -name my_user_clock
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

remove_input_delay

Removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove.

```
remove_input_delay -clock clock_name port_pin_list
remove_input_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified input delay value is assigned.

port_pin_list

Specifies the port names to which the specified input delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the input_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the input_delay constraint ID .

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, RTAX-S, eX (for analysis), and SX-A (for analysis)

Description

Removes an input delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an input delay constraint in the current scenario, or if the specified ID does not refer to an input delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the input delay from CLK1 on port data1:

```
remove_input_delay -clock [get_clocks CLK1] [get_ports data1]
```

See Also

set_input_delay

Tcl Command Documentation Conventions

remove_library

Removes a VHDL library from your project.

```
remove_library
-library name
```

Arguments

`-library name`

Specifies the name of the library you wish to remove.

Supported Families

All

Exceptions

- None

Example

Remove (delete) a library called 'my_lib'.

```
remove_library -library my_lib
```

See Also

add_library

rename_library

remove_max_delay

Removes a maximum delay constraint from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_max_delay [-from from_list] [-to to_list] [-through through_list]
remove_max_delay -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the maximum delay constraint to remove from the current scenario. You must specify either the exact maximum delay arguments to remove or the constraint ID that refers to the maximum delay constraint to remove.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

Removes a maximum delay value from the specified clock in the current scenario. If the arguments do not match a maximum delay constraint in the current scenario, or if the specified ID does not refer to a maximum delay constraint, this command fails.

Do not specify both the maximum delay arguments and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an Accessor command.

Examples

The following example specifies a range of maximum delay constraints to remove:

```
remove_max_delay -through U0/U1:Y
```

See Also

set_max_delay

Tcl Command Documentation Conventions

remove_min_delay

Removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_min_delay [-from from_list] [-to to_list] [-through through_list]
remove_min_delay -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-toto_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the minimum delay constraint to remove from the current scenario. You must specify either the exact minimum delay arguments to remove or the constraint ID that refers to the minimum delay constraint to remove.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

Removes a minimum delay value from the specified clock in the current scenario. If the arguments do not match a minimum delay constraint in the current scenario, or if the specified ID does not refer to a minimum delay constraint, this command fails.

Do not specify both the minimum delay arguments and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example specifies a range of minimum delay constraints to remove:

```
remove_min_delay -through U0/U1:Y
```

See Also

set_min_delay

Tcl Command Documentation Conventions

remove_multicycle_path

Removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_multicycle_path [-from from_list] [-to to_list] [-through through_list]
remove multicycle_path -id constraint_ID
```

Arguments

`-from` ***`from_list`***

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` ***`through_list`***

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`***`to_list`***

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` ***`constraint_ID`***

Specifies the ID of the multicycle path constraint to remove from the current scenario. You must specify either the exact multicycle path arguments to remove or the constraint ID that refers to the multicycle path constraint to remove.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

Description

Removes a multicycle path from the specified clock in the current scenario. If the arguments do not match a multicycle path constraint in the current scenario, or if the specified ID does not refer to a multicycle path constraint, this command fails.

Do not specify both the multicycle path arguments and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes all paths between reg1 and reg2 to 3 cycles for setup check.

```
remove_multicycle_path -from [get_pins {reg1}] -to [get_pins {reg2}]
```

See Also

set_multicycle_path

Tcl Command Documentation Conventions

remove_output_delay

Removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove.

```
remove_output_delay -clock clock_name port_pin_list
remove_output_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified output delay value is assigned.

port_pin_list

Specifies the port names to which the specified output delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the output_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the output_delay constraint ID .

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

Description

Removes an output delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an output delay constraint in the current scenario, or if the specified ID does not refer to an output delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

- You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the output delay from CLK1 on port out1:

remove_output_delay -clock [get_clocks CLK1] [get_ports out1]

See Also

set_output_delay

Tcl Command Documentation Conventions

rename_library

Renames a VHDL library in your project.

```
rename_library
-library name
-name name
```

Arguments

`-library name`

Identifies the current name of the library that you wish to rename.

`-name name`

Specifies the new name of the library.

Supported Families

All

Exceptions

- None

Example

Rename a library from 'my_lib' to 'test_lib1'

```
rename_library -library my_lib -name test_lib1
```

See Also

add_library

remove_library

rename_scenario

Renames the specified timing scenario with the new name provided. You must provide a unique new name (that is, it cannot already be used by another timing scenario).

```
rename_scenario oldname -new newname
```

Arguments

oldname

Specifies the current name of the timing scenario.

`-new newname`

Specifies the new name to give to the timing scenario.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Description

This command changes the name of the timing scenario in the list of scenarios.

Example

```
rename_scenario scenario_A -new scenario_B
```

See Also

create_scenario

delete_scenario

Tcl documentation conventions

Report

The report command provides you with frequently-used information in a convenient format.

You can generate several different types of reports using this command, including:

- report (Status)
- report (Timing) using Timer; for the SX, MX, 3200DX and ACT families
- report (Timing violations) using Timer; for the SX, MX, 3200DX and ACT families
- report (Timing) using SmartTime; for IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, eX, and SX-A families
- report (Timing violations) using SmartTime; for IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, eX, and SX-A families
- report (Pin)
- report (Flip-flop)
- report (I/O Bank)
- report (Global Usage)
- report (Power)

Report (Activity and Hazards Power report)

The activity and hazards report reads a VCD file and reports transitions and hazards for each clock cycle of the VCD file.

```
report -type power_activity_map \
[-vcd_file {path}] \
[-style {value}] \
[-partial_parse {value}] \
[-start_time {value}] \
[-end_time {value}] \
[-auto_detect_top_level_name {value}] \
[-top_level_name {name}] \
[-report_type {value}] \
[-report_query {value}] \
[-sortby {value }] \
[-sortorder {value }] \
[-max_net {value}] \
[-max_cycle {value}] \
[-clock_settings {value}] \
[-glitch_filtering {value}] \
[-glitch_threshold {value}] \
[-auto_construct_clock_domain {value}] \
[-clock_period {value}] \
[-clock_offset {value}] \
[-opmode {value}] \
{filename}
```

Arguments

`-type power_activity_map`

Specifies the type of report to generate is an activity and hazards power report.

`-vcd_file {path}`

Specifies the path to the *.vcd file that you want to import.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

-partial_parse {*value*}

Specifies whether to partially parse the *.vcd file. The following table shows the acceptable values for this argument:

Value	Description
true	Partially parses the *.vcd file
false	Does not partially parse the *.vcd file

-start_time {*value*}

This option is available only if -partially_parse is set to *true*. Specifies the start time (in ns) to partially parse the *.vcd file.

-end_time {*value*}

This option is available only if -partially_parse is set to *true*. Specifies the end time (in ns) to partially parse the *.vcd file.

-auto_detect_top_level_name {*value*}

Specifies whether to automatically detect the top-level name. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the top-level name
false	Does not automatically detect the top-level name

-top_level_name {*name*}

Specifies the top-level name.

-report_type {*value*}

Specifies the report query type. The following table shows the acceptable values for this argument:

Value	Description
activity	Includes activity information for each net
power	Includes power information for each net
activity and power	Includes activity and power information for each net

-report_query {*value*}

Specifies the report type. The following table shows the acceptable values for this argument:

Value	Description
Report by Net - summary	Provides a summary report for each net
Report by Net - detailed	Provides a detailed report for each net
Report by Cycle - summary	Provides a summary report for each cycle
Report by cycle - detailed	Provides a detailed report for each cycle

-sortby {*value*}

Specifies how to sort the values in the report. The following table shows the acceptable values for this argument:

Value	Description
total power	Sorts based on the power values
spurious power	Sorts based on the spurious power
functional power	Sorts based on the functional power
spurious transitions	Sorts based on the spurious transitions
functional transitions	Sorts based on the functional transitions

-sortorder {*value*}

Specifies the sort order of the values in the report. This could be descending or ascending.

-max_net {*value*}

Specifies the maximum number of nets to report. In a net summary or net details report, this argument limits the total number of entries. In a cycle details report, this argument limits the number of nets reported for each cycle.

-max_cycle {*value*}

Specifies the maximum number of cycles to report. In a cycle summary or cycle details report, this argument limits the total number of entries. In a net details report, this argument limits the number of cycles reported for each net

-clock_settings {*value*}

Specifies the settings for the clock. The format is *"< clock name >:< active edge { value } >"*. The following table shows the acceptable values for the active edge:

Value	Description
rising	Sets the clock to a rising active edge
falling	Sets the clock to a falling active edge
both	Sets the clock to both rising and falling active edge
not_active	Does not use the signal as a clock

-glitch_filtering {*value*}

Specifies whether to use glitch filtering. The following table shows the acceptable values for this argument:

Value	Description
true	Glitch filtering is on
auto	Enables automatic glitch filtering. This option will ignore any value specified in -glitch_threshold
false	Glitch filtering is off

-glitch_threshold {*value*}

This option is only available when -glitch_filtering is set to *true*. Specifies the glitch filtering value in ps.

-auto_construct_clock_domain {*value*}

Specifies whether to automatically construct the clock domain. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically constructs the clock domain
false	Does not automatically construct the clock domain

-clock_period {*value*}

Use this option to specify a virtual clock period (in ps). This should be used if -auto_construct_clock_domain is set to *false*.

-clock_offset {*value*}

Use this option to specify the time of the first active edge of the virtual clock (in ps). This should be used if -auto_construct_clock_domain is set to *false*.

-opmode {*value*}

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

{*filename*}

Specifies the name of the report.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example generates an activity and hazards power report named *report_power_activity_map.txt*.

```
report -type "power_activity_map" -vcd_file "D:/FPU/mul.vcd" -style "Text" -
partial_parse "TRUE" -start_time "0.05" -end_time "1.00" -auto_detect_top_level_name
"TRUE" -report_query "Report by Net - Summary" -clock_settings
"UUT/un3_out_3:Y:not_active" -clock_settings "clk:rising" -glitch_filtering "FALSE" -
glitch_threshold "100" -auto_construct_virtual_clock "TRUE" -virtual_clock_period
"10000.00" -virtual_clock_first_edge "0.00" -opmode "Active"\
{D:/FPU/report_power_activity_map.txt}
```

Report (Bottleneck) Using SmartTime

Creates a bottleneck report.

```
report -type bottleneck
[-cost_type {value} ]
[-use_slack_threshold{value} ]
[-slack_threshold {value} ]
[-set_name {value} ]
[-clock clock_id -set_type value ]
[-source_clock clock_id -sink_clock clock_id]
[-source {pin_list} ]
[-sink {pin_list} ]
[-max_instances {value} ]
[-max_paths {value} ]
[-max_parallel_paths {value} ]
[-analysis_type {value} ]
{filename} \
[-format value]
```

Arguments

`-cost_type value`

Specifies the type of bottleneck cost. The default option is path_count.

Value	Description
path_count	Instances with the greatest number of path violations will have the highest bottleneck cost
path_cost	Instances with the largest combined timing violations will have the highest bottleneck cost

`-use_slack_threshold value`

Specifies whether to consider the slack threshold when computing the bottlenecks in the report.

Value	Description
yes	Includes slack threshold in the bottleneck report
no	Excludes slack threshold in the bottleneck report

`-slack_threshold value`

Specifies that paths whose slack is larger than this given threshold will be considered. Only instances that lie on these violating paths are reported. The default option is 0.

`-set_name value`

Displays the bottleneck information for the named set. You can either use this option or use both –clock and –type. This option allows pruning based on a given set. Only paths that lie within the named set will be considered towards bottleneck.

`-clock value`

This option allows pruning based on a given clock domain. Only instances that lie on these violating paths are reported.

`-set_type value`

This option can only be used in combination with the –clock option, and not by itself. The options allow to filter which type of paths should be considered towards the bottleneck.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock clock_id`

Reports only bottleneck instances that lie on violating timing paths of the inter-clock domain that starts at the source clock specified by this option. This option can only be used in combination with -sink_clock, and not by itself.

`-sink_clock clock_id`

Reports only bottleneck instances that lie on violating timing paths of the inter-clock domain that ends at the sink clock specified by this option. This option can only be used in combination with -source_clock, and not by itself.

`-source value`

Reports only instances that lie on violating paths that start at locations specified by this option.

`-sink value`

Reports only instances that lie on violating paths that end at locations specified by this option.

`-max_instances value`

Specifies the maximum number of instances to be reported. Defaults to 10.

`-max_paths value`

Specifies the maximum number of paths to be considered per path set type. Allowed values are 1 to 2000000. Defaults to 100.

`-max_parallel_paths value`

Specifies the maximum number of paths allowed per end point pair. Only instances that lie on these violating paths are reported. Defaults to 1 (No parallel paths).

`-analysis_type value`

Specifies the analysis types (max or min) under which the violations are reported. Defaults to max analysis.

Value	Description
max_delay	Sets the analysis type to maximum delay
min_delay	Sets the analysis type to minimum delay

`-format value`

Specifies the output format of the generated report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format that you can import into a spreadsheet

filename

Specifies the name and destination of the bottleneck report.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Exceptions

None

Examples

The following example generates a bottleneck report named bottleneck.txt.

```
report -type bottleneck -cost-type path_count -slack_threshold 0 -set_name set1 -
max_cells 10 -max_paths 10 -max_parallel_paths 10 -analysis_type max -format text
bottleneck.txt
```

See Also

Tcl documentation conventions

Report (data change history)

Creates a Data Change History report, which lists new features and enhancements, bug fixes and known issues for the current release that may impact the power consumption of the design.

```
report -type power_history \
{report.txt}
```

Arguments

`-type power_history`

Specifies the type of report to generate is a data change history report.

`{report.txt}`

Specifies the name of the report. You must use .txt as the filename extension.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example generates a data change history report named *report.txt*.

```
report -type "power_history" \
 {report.txt}
```

Report (Cycle Accurate Power report)

Creates a cycle accurate power report, which reports a power waveform with one power value per clock period or half-period instead of an average power for the whole simulation.

```
report -type power_peak_analyzer \
[-vcd_file {path}] \
[-style {value}] \
[-partial_parse {value}] \
[-start_time {value}] \
[-end_time {value}] \
[-auto_detect_top_level_name {value}] \
[-top_level_name {name}] \
[-glitch_filtering {value}] \
[-glitch_threshold {value}] \
[-auto_detect_sampling_period {value}] \
[-sampling_clock { }] \
[-sampling_rate_per_period {value}] \
[-sampling_offset {value}] \
[-sampling_period {value}] \
[-use_only_local_extrema {value}] \
[-use_power_threshold {value}] \
[-power_threshold {value}] \
[-opmode {value}] \
{filename}
```

Arguments

`-type power_peak_analyzer`

Specifies the type of report to generate is a cycle accurate power report.

`-vcd_file {path}`

Specifies the path to the *.vcd file that you want to import.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-partial_parse {value}`

Specifies whether to partially parse the *.vcd file. The following table shows the acceptable values for this argument:

Value	Description
true	Partially parses the *.vcd file
false	Does not partially parse the *.vcd file

`-start_time {value}`

This option is available only if `-partially_parse` is set to *true*. Specifies the start time (in ns) to partially parse the *.vcd file.

`-end_time {value}`

This option is available only if -partially_parse is set to *true*. Specifies the end time (in ns) to partially parse the *.vcd file.

-auto_detect_top_level_name {*value*}

Specifies whether to automatically detect the top-level name. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the top-level name
false	Does not automatically detect the top-level name

-top_level_name {*name*}

Specifies the top-level name.

-glitch_filtering {*value*}

Specifies whether to use glitch filtering. The following table shows the acceptable values for this argument:

Value	Description
true	Glitch filtering is on
auto	Enables automatic glitch filtering. This option will ignore any value specified in -glitch_threshold
false	Glitch filtering is off

-glitch_threshold {*value*}

This option is only available when -glitch_filtering is set to *true*. Specifies the glitch filtering value (in ps).

-power_summary {*value*}

Specifies whether to include the power summary, which shows the static and dynamic values in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power summary in the report
false	Does not include the power summary in the report

`-auto_detect_sampling_period {value}`

Specifies whether to automatically detect the sampling period. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the sampling period
false	Does not automatically detect the sampling period

`-sampling_clock {}`

Specifies the sampling clock.

`-sampling_rate_per_period {value}`

Specifies whether to set the sampling rate per period. The following table shows the acceptable values for this argument:

Value	Description
true	Specifies the sampling rate per period
false	Specifies the sampling rate per half period

`-sampling_offset {value}`

Specifies the offset used to calculate the sampling offset (in ps).

`-sampling_period {value}`

Specifies the offset used to calculate the sampling period (in ps).

`-use_only_local_extrema {value}`

Specifies whether to limit the history size by keeping only local extrema. The following table shows the acceptable values for this argument:

Value	Description
true	Limits the history size by keeping only local extrema
false	Does not limit the history size by keeping only local extrema

`-use_power_threshold {value}`

Specifies whether to limit the history size by setting a power threshold. The following table shows the acceptable values for this argument:

Value	Description
true	Limits the history size by setting a power threshold
false	Does not limit the history size by setting a power threshold

-power_threshold {*value*}

Sets the power threshold value.

-opmode {*value*}

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

{*filename*}

Specifies the name of the report.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example generates a cycle accurate power report named report_power_cycle_based.txt.

```
report -type "power_cycle_based" -vcd_file "D:/FPU/mul.vcd" -style "Text" -partial_parse
"TRUE" -start_time "0.05" -end_time "1.00" -auto_detect_top_level_name "TRUE" -
glitch_filtering "FALSE" -glitch_threshold "100" -auto_detect_sampling_period "TRUE" -
sampling_clock "clk" -sampling_rate_per_period "TRUE" -sampling_offset "0.00" -
sampling_period "10000.00" -use_only_local_extrema "TRUE" -use_power_threshold "TRUE" -
power_threshold "0.00" -opmode "Active" \ {D:/FPU/report_power_cycle_based.txt}
```

Report (datasheet) Using SmartTime

Creates a datasheet report.

```
report -type datasheet filename \
[-format value]
```

Arguments

filename

Specifies the name and destination of the datasheet report.

-format *value*

Specifies the output format of the generated the report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, Axcelerator, eX, and SX-A

Exceptions

None

Examples

The following example generates a datasheet report named datasheet.txt.

```
report -type datasheet -format Text datasheet.txt
```

See Also

Tcl documentation conventions

report (Timing) using SmartTime

report (Timing violations) using SmartTime

Report (Power)

Creates a Power report, which enables you to determine if you have any power consumption problems in your design. It includes information about the global device and SmartPower preferences selection, and hierarchical detail (including gates, blocks, and nets), with a block-by-block, gate-by-gate, and net-by-net power summary SmartPower results.

```
report -type power \
[-powerunit {value}] \
[-frequnit {value}] \
[-opcond {value}] \
[-opmode {value}] \
[-toggle {value}] \
[-power_summary {value}] \
[-rail_breakdown{value}] \
[-type_breakdown{ value}] \
[-clock_breakdown{value}] \
[-thermal_summary {value}] \
[-battery_life {value}] \
[-opcond_summary {value}] \
[-clock_summary {value}] \
[-style {value}] \
[-sortorder {value}] \
[-sortby {value}] \
[-instance_breakdown {value}] \
[-power_threshold {value}] \
[-filter_instance {value}] \
[-min_power {number}] \
[-max_instance {integer >= 0}] \
[-activity_sortorder {value}] \
[-activity_sortby {value}] \
[-activity_summary {value}] \
[-frequency_threshold {value}] \
```

```
[-filter_pin {value}] \
[-min_frequency {value}] \
[-max_pin {value}] \
[-enablerates_sortorder {value}] \
[-enablerates_sortby {value}] \
[-enablerates_summary {value}] \
{filename}
```

Arguments

`-type power`

Specifies the type of report to generate is a Power report.

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
active	The operating mode is set to active
static	The operating mode is set to static
sleep	The operating mode is set to sleep
Flash*Freeze	The operating mode is set to Flash*Freeze
shutdown	The operating mode is set to shutdown

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

`-power_summary {value}`

Specifies whether to include the power summary, which shows the static and dynamic values in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power summary in the report
false	Does not include the power summary in the report

`-rail_breakdown {value}`

Specifies whether to include the breakdown by rail summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by rail summary in the report
false	Does not include the breakdown by rail summary in the report

`-type_breakdown {value}`

Specifies whether to include the breakdown by type summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by type summary in the report
false	Does not include the breakdown by type summary in the report

`-clock_breakdown {value}`

Specifies whether to include the breakdown by clock domain in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by clock domain summary in the report
false	Does not include the breakdown by clock domain summary in the report

`-thermal_summary {value}`

Specifies whether to include the thermal summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the thermal summary in the report
false	Does not include the thermal summary in the report

`-battery_life {value}`

Specifies whether to include the battery life summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the battery life summary in the report
false	Does not include the battery life summary in the report

`-opcond_summary {value}`

Specifies whether to include the operating conditions summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the operating conditions summary in the report
false	Does not include the operating conditions summary in the report

`-clock_summary {value}`

Specifies whether to include the clock domains summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the clock summary in the report
false	Does not include the clock summary in the report

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-sortby {value}`

Specifies how to sort the values in the report. The following table shows the acceptable values for this argument:

Value	Description
power values	Sorts based on the power values
alphabetical	Sorts in an alphabetical order

`-sortorder {value}`

Specifies the sort order of the values in the report. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-instance_breakdown {value}`

Specifies whether to include the breakdown by instance in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by instance in the report
false	Does not include the breakdown by instance in the report

`-power_threshold {value}`

This specifies whether to include only the instances that consume power above a certain minimum value. When this command is set to true, the `-min_power` argument must also be used to specify that only the instances that consume power above this minimum power value are the ones that are included in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power threshold in the report
false	Does not include the power threshold in the report

`-filter_instance {value}`

This specifies whether to have a limit on the number of instances to include in the Power report. When this command is set to true, the `-max_instance` argument must also be used to specify the maximum number of instances to be included into the Power report. The following table shows the acceptable values for this argument:

Value	Description
true	Indicates that you want to have a limit on the number of instances to include in the Power report
false	Indicates that you do not want to have a limit on the number of instances to include in the Power report

`-min_power {number}`

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

`-max_instance {integer >= 0}`

Sets the maximum number of instances to a specified integer greater than or equal to 0 (zero). This will limit the maximum number of instances to be included in the Power report.

`-activity_sortorder {value}`

Specifies the sort order for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

-activity_sortby {*value*}

Specifies how to sort the values for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the power values
domain	Sorts based on the clock domain
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

-activity_summary {*value*}

Specifies whether to include the activity summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

-frequency_threshold {*value*}

Specifies whether to add a frequency threshold. The following table shows the acceptable values for this argument:

Value	Description
true	Adds a frequency threshold
false	Does not add a frequency threshold

-filter_pin {*value*}

Specifies whether to filter by maximum number of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Filters by maximum number of pins
false	Des not filter by maximum number of pins

`-min_frequency {value}`

Sets the minimum frequency to {decimal value [unit { Hz | KHz | MHz}]}.

`-max_pin {value}`

Sets the maximum number of pins.

`-enablerates_sortorder {value}`

Specifies the sort order for the enable rates summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-enablerates_sortby {value}`

Specifies how to sort the values for the enable rates summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the power values
domain	Sorts based on the clock domain
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

`-enablerates_summary {value}`

Specifies whether to include the enable rates summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

`{filename}`

Specifies the name of the report.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- The following arguments have been removed. Running the script will trigger a warning message: Warning: Invalid argument: -argname "argvalue" Ignored. Ignore the warning.

```
-annotated_pins {value}
-stat_pow {value}
-dyn_pow {value}
```

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best are available only for certain families and devices.

Exceptions

- None

Examples

This example generates a Power report named report.rpt.

```
report -type "power" \
 "Power Values" \
 "Descending" \
"TRUE" \
 "FALSE" \
 "FALSE" \
 "Typical" \
 -min_power "2 mW" \
 "ACTIVE" \
 "TRUE" \
 "TRUE" \
 "TRUE" \
 "TRUE" \
 "5" \
 "TRUE" \
 {e:\SmartPower\report.rpt}
```

Report (Power Scenario)

Creates a scenario power report for a previously defined scenario. It includes information about the global device and SmartPower preferences selection, and the average power consumption and the excepted battery life for this sequence.

```
report -type power_scenario \
[-powerunit {value}] \
[-frequnit {value}] \
[-opcond {value}] \
[-toggle {value}] \
[-scenario {value}] \
[-style {value}] \
[-battery_life {value}] \
[-battery_capacity {value}] \
[-rail_breakdown {value}] \
[-type_breakdown {value}] \
[-mode_breakdown {value}] \
[-opcond_summary {value}] \
{filename}
```

Arguments

`-type power_scenario`

Specifies the type of report to generate is a scenario power report.

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

`-scenario{value}`

Specifies a scenario that the report is generated from.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-battery_life {value}`

Specifies whether to include the battery life summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the battery life summary in the report
false	Does not include the battery life summary in the report

`-battery_capacity {value}`

Specifies the battery capacity in A*H.

`-rail_breakdown {value}`

Specifies whether to include the breakdown by rail summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by rail summary in the report
false	Does not include the breakdown by rail summary in the report. This is the default value.

-type_breakdown {*value*}

Specifies whether to include the breakdown by type summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by type summary in the report
false	Does not include the breakdown by type summary in the report. This is the default value.

-mode_breakdown {*value*}

Specifies whether to include a breakdown by mode in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by mode in the report
false	Does not include the breakdown by mode in the report. This is the default value.

-opcond_summary {*value*}

Specifies whether to include the operating conditions summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the operating conditions summary in the report
false	Does not include the operating conditions summary in the report

{filename.rpt}

Specifies the name of the report.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best are available only for certain families and devices.

Exceptions

- None

Examples

This example generates a scenario power report named report.txt for my_scenario

```
report -type power_scenario -scenario my_scenario -rail_breakdown true -type_breakdown
true -mode_breakdown true -style text -battery_capacity 10 report.txt
```

See Also

Scenario Power Report

Report (Timing) Using SmartTime

Creates a timing report.

```
report -type timing filename\
[-print_summary value]\
[-analysis value]\
[-use_slack_threshold value]\
[-slack_threshold value]\
[-print_paths value]\
[-max_paths value]\
[-max_expanded_paths value]\
[-include_user_sets value]\
[-include_pin_to_pin value]\
[-select_clock_domains value]\
[-clock_domain clock_domain_list]
[-format value]
```

Arguments

filename

Specifies the name and destination of the timing report.

`-type timing`

Specifies the type of report to generate.

`-print_summary` *value*

Specifies whether to print the summary section in the timing report.

Value	Description
yes	Includes summary section in the timing report (the default value).
no	Excludes summary section in the timing report

`-analysis` *value*

Specifies whether the report will consider minimum analysis or maximum analysis.

Value	Description
min	Timing report considers minimum analysis
max	Timing report considers maximum analysis (the default value)

`-use_slack_threshold` *value*

Specifies whether the report will consider slack threshold.

Value	Description
yes	Includes slack threshold in the timing report.
no	Excludes slack threshold in the timing report (the default value)

`-slack_threshold` *value*

Specifies the threshold to consider when reporting path slacks. This is a floating-point number in nanoseconds (ns). By default, there is no threshold (all slacks are reported).

`-print_paths` *value*

Specifies whether the path section (clock domains and in-to-out paths) will be printed in the timing report.

Value	Description
yes	Includes path section in the timing report (the default value)
no	Excludes path sections from the timing report

`-max_paths` *value*

Defines the maximum number of paths to display for each set. This is a positive integer value greater than zero. The default is 5.

`-max_expanded_paths` *value*

Defines the number of paths to expand per set. This is a positive integer value greater than zero. The default is 1.

`-include_user_sets` *value*

Defines whether to include the user defined sets in the timing report.

Value	Description
yes	Includes user defined sets in the timing report (the default value)
no	Excludes user defined sets from the timing report

`-include_pin_to_pin` *value*

Specifies whether to show pin-to-pin paths in the timing report.

Value	Description
yes	Includes pin-to-pin paths in the timing report (the default value).
no	Excludes pin-to-pin paths from the timing report

-select_clock_domains *value*

Specifies whether to show the clock domain list in the timing report.

Value	Description
yes	Includes the clock domain list in the timing report
no	Excludes the clock domain list from the timing report (the default value)

-clock_domain clock_domain_list

Defines the clock domain to be considered in the clock domain section. The domain list is a series of strings with domain names separated by spaces. Both the summary and the path sections in the timing report display only the listed clock domains.

-format *value*

Specifies the output format of the generated the report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Exceptions

None

Examples

The following example generates a timing report named timing_report.txt. The report does not print the summary section. It includes a max-delay analysis and only reports paths with a slack value less than 0.50 ns. It reports a maximum of 3 paths per section and does not report any expanded paths. It only reports timing information for the clock domains count8_clock and count2_clk.

```
report -type timing -print_summary no \
-analysis max \
-use_slack_threshold yes \
-slack_threshold 0.50 \
-print_paths yes -max_paths 3 \
-max_expanded_paths 0 \
-include_user_sets yes \
-include_pin_to_pin yes \
-select_clock_domains yes \
```

```
-clock_domain {count8_clock count2_clk} \
timing_report.txt
```

See Also

Tcl documentation conventions

report (Timing violations) using SmartTime

report (Datasheet) using SmartTime

Report (timing violations) Using SmartTime

Creates a timing violations report.

```
report -type timing_violations filename  \
[-analysis value]\
[-use_slack_threshold value]\
[-slack_threshold value]\
[-limit_max_paths value]\
[-max_paths value]\
[-max_expanded_paths value] \
[-format value]
```

Arguments

filename

Specifies the name and destination of the timing violations report.

-type timing_violations

Specifies the type of report to generate.

-analysis *value*

Specifies whether to consider minimum analysis or maximum analysis in the timing violations report.

Value	Description
min	Timing report considers minimum analysis
max	Timing report considers maximum analysis (the default value)

-use_slack_threshold *value*

Specifies whether to consider the slack threshold in the timing violations report.

Value	Description
yes	Includes slack threshold in the timing violations report
no	Excludes slack threshold in the timing violations report (the default value)

-slack_threshold *value*

Specifies the threshold to consider when reporting path slacks. This value is a floating-point number in nanoseconds (ns). By default, there is no threshold (all slacks reported).

-limit_max_paths *value*

Specifies if the paths are limited by the number of paths.

Value	Description
yes	Limits the maximum number of paths to report
no	Specifies that there is no limit to the number of paths to report (the default value)

-max_paths *value*

Specifies the maximum number of paths to display for each set. This value is a positive integer value greater than zero. Default is 100.

-max_expanded_paths *value*

Specifies the number of paths to expand per set. This value is a positive integer value greater than zero. The default is 0.

-format *value*

Specifies the output format of the generated report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

The following example generates a timing violations report named timg_viol.txt. The report considers an analysis using maximum delays and does not filter paths based on slack threshold. It reports 2 paths per section and 1 expanded path per section.

```
report -type timing_violations \
-analysis max -use_slack_threshold no \
-limit_max_paths -yes \
-max_paths 2 \
-max_expanded_paths 1 \
timg_viol.txt
```

See Also

Tcl documentation conventions

report (Timing) using SmartTime

report (Datasheet) using SmartTime

save_design

The save_design command saves the current design in Designer to a file. If filename is not a complete path name, the ADB file is written into the current working directory.

```
save_design filename
```

Arguments

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

Supported Families

All

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

close_design

new_design

open_design

set_clock_latency

Defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies the source latency on a clock pin, potentially only on certain edges of the clock.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`delay`

Specifies the latency value for the constraint.

`clock`

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S,eX, and SX-A

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Exceptions

- None

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency -source -rise -early 0.4 { main_clock  }
set_clock_latency -source -fall    1.2 { main_clock }
```

See Also

create_clock

create_generated_clock

Tcl Command Documentation Conventions

set_current_scenario

Specifies the timing scenario for the Timing Analyzer to use. All commands that follow this command will apply to the specified timing scenario.

```
set_current_scenario name
```

Arguments

name

Specifies the name of the timing scenario to which to apply all commands from this point on.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Description

A timing scenario is a set of timing constraints used with a design. If the specified scenario is already the current one, this command has no effect.

After setting the current scenario, constraints can be listed, added, or removed, the checker can be invoked on the set of constraints, and so on.

This command uses the specified timing scenario to compute timing analysis.

Exceptions

- None

Example

```
set_current_scenario scenario_A
```

See Also

get_current_scenario

Tcl Command Documentation Conventions

set_defvar

The set_defvar command sets an internal variable in the Designer system. You must specify at least one argument for this command.

```
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.

Supported Families

All

Exceptions

None.

Example

Example 1:

```
set_defvar "FORMAT" "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 variables for this command.

See Also

get_defvar

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.

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

Note: You need all three arguments for this command to set up your design.

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.

Supported Families

All

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. You must specify at least one option for this command. Some of the options may not apply for certain families that do not support the features.

```
set_device -family family_name -die die_name -package package_name -speed speed_grade -voltage
voltage -voltrange volt_range -temprange temp_range -iostd default_io_std -pci value -jtag value
-probe value -trst value -radexp value
```

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. You can also use it to define the I/O voltage of the part. For example, if you are using a RTSX with a 3.3 to 2.5 voltage, you can use

-voltage 3.3/2.5

`-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 temperature range to be applied for the device. Temperature rantes are MIL, COM and IND denoting Military, Commercial and Industrial respectively. Automotive applications generally use the Automotive, TGrade1, or TGrade2 temperature range.

`-iostd default_io_std`

Specifies the default I/O standard of the part.

`-pci value`

Used if the device needs to configure the I/Os for PCI specification. This parameter is equivalent to setting your I/O attributes to PCI in the Device Selection Wizard. Values are summarized in the table below.

Value	Description
yes	Device is configured for PCI specification
no	Device is not configured for PCI specification

-jtag *value*

Specifies if pins need to be reserved for JTAG. Values are summarized in the table below.

Value	Description
yes	Pins are reserved for JTAG
no	Pins are not reserved for JTAG

-probe *value*

Specifies if the pins need to be preserved for probing. Values are summarized in the table below.

Value	Description
yes	Pins are preserved for probing
no	Pins not preserved for probing

-trst *value*

Specifies if the pins need to be reserved for JTAG test reset. Values are summarized in the table below.

Value	Description
yes	Pins are preserved for JTAG test reset
no	Pins are not preserved for JTAG test reset

-radexp *value*

Specifies the radiation value (in Krad) for radiation tolerant devices.

Supported Families

All

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_disable_timing

Disables timing arcs within a cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing -from value -to value name
```

Arguments

`-from from_port`

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to to_port`

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the timing arcs will be disabled.

Supported Families

IGLOO, Fusion, ProASIC3, Axcelerator, and RTAX-S

Exceptions

- None

Example

```
set_disable_timing -from A -to Y a2
```

set_false_path

Identifies paths that are considered false and excluded from the timing analysis in the current timing scenario.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path -from [get_clocks {clk1}] -to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

See Also

Tcl Command Documentation Conventions

set_input_delay

Creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario.

```
set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] input_list
```

Arguments

`delay_value`

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock` *`clock_ref`*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`input_list`

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 -min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 -max {IN1}
```

See Also

set_output_delay

Tcl Command Documentation Conventions

set_max_delay

Specifies the maximum delay for the timing paths in the current scenario.

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

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual maximum delay targets from clock waveforms and port input or output delays.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

See Also

set_min_delay

remove_max_delay

Tcl Command Documentation Conventions

set_min_delay

Specifies the minimum delay for the timing paths in the current scenario.

```
set_min_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC (for analysis), Axcelerator, RTAX-S, eX (-through option), and SX-A (-through option)

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual minimum delay targets from clock waveforms and port input or output delays.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

See Also

set_max_delay

remove_min_delay

Tcl Command Documentation Conventions

set_multicycle_path

Defines a path that takes multiple clock cycles in the current scenario.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list[-through through_list[-to
to_list
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

-setup

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

-hold

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

You must specify at least one of the -from, -to, or -through arguments for this constraint to be valid.

Exceptions

- Multiple priority management is not supported in Actel designs. All multiple cycle path constraints are handled with the same priority.

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] -to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

See Also

remove_multicycle_path

Tcl Command Documentation Conventions

set_output_delay

Defines the output delay of an output relative to a clock in the current scenario.

```
set_output_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

Arguments

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

-max

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-min

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 -min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK2 -max {OUT1}
```

See Also

remove_output_delay

set_input_delay

Tcl Command Documentation Conventions

smartpower_add_new_custom_mode

Creates a new custom mode.

```
smartpower_add_new_custom_mode -name {mode_name} -base_mode {base_mode} -description
{mode_description}
```

Arguments

-name {*mode_name*}

Specifies the name of the new custom mode.

-base_mode {*base_mode*}

Specifies the name of the base mode used to create the new custom mode.

-description {*mode_description*}

Specifies the description of the new custom mode.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example creates a new custom mode "Cust_1" based on the Active mode:

```
smartpower_add_new_custom_mode -name {Cust_1} -base_mode {Active} -description
{frequency 10 MHz}
```

See Also

smartpower_remove_custom_mode

smartpower_add_new_scenario

Creates a new scenario.

```
smartpower_add_new_scenario -name {value} -description {value} -mode {value}
```

Arguments

`-name {value}`

Specifies the name of the new scenario.

`-description {value}`

Specifies the description of the new scenario.

`-mode {<operating mode>:<duration>}+`

Specifies the mode(s) and duration(s) for the specified scenario.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC PLUS, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example creates a new scenario called myscenario:

```
smartpower_add_new_scenario -name myscenario -description mynewscenario -mode active:30
+ shutdown:30 + active:40
```

smartpower_add_pin_in_domain

Adds a pin into a clock or set domain.

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type {value} -domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to add to the domain.

`-pin_type {value}`

Specifies the type of the pin to add. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to add is a clock pin
data	The pin to add is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain in which to add the specified pin.

`-domain_type {value}`

Specifies the type of domain in which to add the specified pin. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

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

Examples

The following example adds a clock 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}
```

The following example adds 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

Saves the changes to the design (.adb) file.

```
smartpower_commit
```

Arguments

- None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

```
smartpower_commit
```

See Also

smartpower_restore

smartpower_create_domain

Creates a new clock or set domain.

```
smartpower_create_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

Specifies the type of domain to create. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

Specifies the name of the new domain.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- The `domain name` cannot be the name of an existing domain.
- The domain `type` must be either clock or set.

Exceptions

- None

Examples

The following example creates a new clock domain named "clk2":

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

The following example creates a new set domain named "myset":

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

See Also

smartpower_remove_domain

smartpower_edit_custom_mode

Edits a custom mode.

```
smartpower_edit_custom_mode -name {old_mode_name} new_name {new_mode_name} -description
{mode_description}
```

Arguments

`-name {old_mode_name}`

Specifies the name of the custom mode you want to edit.

`-new_name {new_mode_name}`

Specifies the new name of the custom mode.

`-description {mode_description}`

Specifies the description of the new custom mode.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example edits custom mode "Cust_1" and renames it "Cust_2":

```
smartpower_edit_custom_mode -name {Cust_1} -new_name {Cust_2} -description {frequency 10
MHz}
```

See Also

smartpower_remove_custom_mode

smartpower_add_custom_mode

smartpower_edit_scenario

Edits a scenario.

```
smartpower_edit_scenario -name {value} -description {value} -mode {value} -new_name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

`-description {value}`

Specifies the description of the scenario.

`-mode {<operating mode>:<duration>}`

Specifies the mode(s) and duration(s) for the specified scenario.

`-new_name {value}`

Specifies the new name for the scenario

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example edits the name of myscenario to finalscenario:

```
smartpower_edit_scenario -name myscenario -new_name finalscenario
```

smartpower_initialize_allclocks

Initializes the clock frequency and the data frequency of all clock domains with the initialization options.

```
smartpower_initialize_allclocks -with_clock_constraints {value} -with_clock_freq {value} -
clock_freq {value} -with_data_freq {value} -data_freq {value}
```

Arguments

`-with_clock_constraints {value}`

This sets the option of initializing the clock frequencies of all clock domains with frequency constraints from SmartTime. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with clock constraints ON
false	Sets initialize clock frequencies with clock constraints OFF

`-with_clock_freq {value}`

This sets the option of initializing the clock frequencies with a user input frequency. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with fixed frequency ON
false	Sets initialize clock frequencies with fixed frequency OFF

`-clock_freq {value}`

Specifies the user input clock frequency in MHz.

`-with_data_freq {value}`

Sets the option of initializing the data frequencies of all clock domains with a user input toggle rate or a user input frequency. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize data frequencies with fixed frequency ON
false	Sets initialize data frequencies with fixed frequency OFF

`-data_freq {value}`

Specifies the user input toggle rate or data frequency. If the value is a percentage number such as 20.0 %, it will be interpreted as a toggle rate. If the value is a decimal number in MHz such as 100.0 MHz, it will be interpreted as a fixed frequency value.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- This command is associated with the functionality of Initialize with SmartTime dialog box.

Exceptions

- None

Examples

The following example initializes all clocks with clock constraints from SmartTime:

```
smartpower_initialize_allclocks -with_clock_constraints {true}
```

smartpower_initialize_clock_with_constraints

Initializes the clock frequency and the data frequency of a single clock domain with a specified clock name and the initialization options.

```
smartpower_initialize_clock_with_constraints -clock_name {value}
```

Arguments

`-clock_name {value}`

Specifies the name of the clock that will be initialized.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- This command is associated with the functionality of Initialize with SmartTime dialog box.
- This command is associated with the right-click menu Synchronize Domain with SmartTime on a single clock domain in the Domains tab.

Exceptions

- None

Examples

The following example initializes "my_clock" with clock constraints from SmartTime:

```
smartpower_initialize_clock_with_constraints -clock_name {my_clock}
```

smartpower_remove_custom_mode

Removes a custom mode.

```
smartpower_remove_custom_mode -name {deleted_mode_name}
```

Arguments

`-name {deleted_mode_name}`

Specifies the name of the custom mode you want to delete.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example delets custom mode "Cust_1":

```
smartpower_delete_custom_mode -name {Cust_1}
```

See Also

sp_add_custom_mode

sp_edit_custom_mode

smartpower_remove_domain

Removes an existing clock or set domain.

```
smartpower_remove_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

This specifies the type of domain to remove.The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

This specifies the name of the domain to remove

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

The domain name must be the name of an existing domain.

The domain type must be either clock or set.

Exceptions

None

Examples

The following example removes the clock domain named "clk2":

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

The following example removes the set domain named "myset":

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

See Also

smartpower_create_domain

smartpower_remove_file

Removes a VCD file from the specified mode.

```
remove_file
[-file {value}] \
[-format {value}] \
[-opmode {value}] \
```

Arguments

`-file {value}`

Specifies the file to be removed.

`-format VCD`

Specifies that the type to be removed is a VCD file.

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
active	The operating mode is set to active
static	The operating mode is set to static
sleep	The operating mode is set to sleep
Flash*Freeze	The operating mode is set to Flash*Freeze
shutdown	The operating mode is set to shutdown

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- Noe

Exceptions

- Nonne

Examples

This example removes the file test.vcd from the active mode.

```
smartpower_remove_file -file test -format -vcd opmode -active
```

smartpower_remove_pin_enable_rate

Removes the enable rate value associated with a specific pin. This pin will have a default enable rate based on the domain set it belongs to.

```
smartpower_remove_pin_enable_rate -pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the enable rate to remove. This pin must be the direct driver of an enable pin.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example removes the enable rate of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_enable_rate -pin_name mybibuf/U0/U1:EOUT
```

See Also

smartpower_set_pin_enable_rate

smartpower_set_default_enable_rate

smartpower_remove_pin_frequency

Removes the frequency associated with a specific pin. This pin will have a default frequency based on its domain.

```
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.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^PLUS, ProASIC, and Axcelerator

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

Examples

The following example removes the frequency from the pin named "count8_clock":

```
smartpower_remove_pin_frequency -pin_name {count8_clock}
```

See Also

smartpower_set_pin_frequency

smartpower_remove_pin_of_domain

Removes a clock pin or a data pin from a clock or set domain, respectively.

```
smartpower_remove_pin_of_domain -pin_name {pin_name} -pin_type {value} -domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to remove from the domain.

`-pin_type {value}`

Specifies the type of the pin to remove. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to remove is a clock pin
data	The pinto remove is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain from which to remove the pin.

`-domain_type {value}`

Specifies the type of domain from which the pin is being removed. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- The domain name must be the name of an existing domain.
- The pin name must be the name of an existing pin.

Exceptions

- None

Examples

The following example removes the clock pin named "XCMP3/UO/U1:Y" from the clock domain named "clockh":

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

The following example removes the data pin named "count2_en" from the set domain named "InputSet":

```
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_remove_scenario

Remmoves a scenario from the current design.

```
smartpower_remove_scenario -name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example removes a scenario from the current design:

```
smartpower_remove_scenario -name myscenario
```

smartpower_remove_vcd

Removes an existing VCD file from a mode or entire design.

```
smartpower_remove_vcd -from {value} -mode {value} -file
```

Arguments

`-from {value}`

This specifies the if the VCD is removed for a specific mode or for the entire project. The following table shows the acceptable values for this argument:

Value	Description
mode	The VCD file is removed for a mode
project	The VCD file is removed from the project

`-mode {value}`

This specifies the name of the mode for which the VCD will be removed

`-filename`

This specifies the name of the VCD file to be removed

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

None

Exceptions

None

Examples

The following example removes the VCD file named *my_vcd.vcd* from the active mode

```
smartpower_remove_vcd -from {mode} -mode {active} -my_vcd.vcd
```

See Also

smartpower_create_domain

smartpower_restore

Restores all power information previously committed in SmartPower.

```
smartpower_restore
```

Arguments

- None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

```
smartpower_restore
```

See Also

smartpower_commit

smartpower_set_battery_capacity

Sets the battery capacity.

```
smartpower_set_battery_capacity {value}
```

Arguments

value

Sets the battery capacity to a value in mA/h.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example sets the battery capacity to 40 A/h:

```
smartpower_set_battery_capacity {40}
```

smartpower_set_cooling

Sets the cooling style to one of the predefined types, or a custom value.

```
smartpower_set_cooling -style {value} -teta {value}
```

Arguments

`-style {value}`

Specifies the cooling style to custom value or to one of the predefined types with a default thermal resistance value. The following table shows the acceptable values for this argument:

Value	Description
300_lfm	Predefined cooling style
case_cooling	Predefined cooling style
still_air	Predefined cooling style
custom	Cooling style defined by user input

`-teta {value}`

Specifies the thermal resistance in °C/W. This argument is only available when style is set to Custom.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- To compute the junction temperature, set the following three commands: smartpower_set_thermalmode, smartpower_set_tambient and smartpower_set_cooling. The junction temperature will be updated when an output command is executed, such as report(Power).

Exceptions

- None

Examples

The following example sets the cooling style to still air:

```
smartpower_set_cooling -style {still_air}
```

smartpower_set_default_enable_rate

Sets the enable-rate of one of the enable set domains: IOEnableSet or MemoriesEnableSet.

```
smartpower_set_default_enable_rate -domain_name {name} pin_enable_rate {value}
```

Arguments

`-domain_name {name}`

Specifies the name of the domain value of which you want to apply the enable rate value. The domain_name can be IOsEnableSet or MemoriesEnableSet.

`-pin_enable_rate {value}`

Specifies the value of the pin enable rate as a percentage, which can be any positive decimal between 0 and 100, inclusive.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example sets the enable rate for all the pins belonging to the domain set IOsEnableSet:

```
smartpower_set_default_enable_rate -domain_name IOsEnableSet -pin_enable_rate 52.2
```

See Also

smartpower_set_pin_enable_rate

smartpower_remove_pin_enable_rate

smartpower_set_domain_frequency

Sets the frequency of a domain in megahertz (MHz).

```
smartpower_set_domain_frequency -domain_type {value} -domain_name {domain_name} -clock_freq {value} -data_freq {value} -pin_freq {value}
```

Arguments

`-domain_type {value}`

Specifies the type of domain to set. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

Specifies the name of the domain for which the frequency will be set.

`-clock_freq {value}`

Specifies the clock frequency in megahertz (MHz), which can be any positive decimal number. This argument is available only for a clock domain.

`-data_freq {value}`

Specifies the data frequency in megahertz (MHz), which can be any positive decimal number. This argument is available only for a clock domain.

`-pin_freq {value}`

Specifies the value of the pin frequency in megahertz (MHz), which can be any positive decimal number, which can be any positive decimal number. This argument is available only for a set domain.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- The domain type must be either *clock* or *set*.
- The domain name must be the 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. You must enter a space between the frequency value and the unit. You set the clock frequency only for clock domains.
- The data frequency must be a positive decimal number. Specifying the unit as part of the data frequency value is optional. You must enter a space between the data frequency value and the unit.

Exceptions

- None

Examples

The following example sets the clock and 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}
```

The following example sets 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_mode_for_analysis

Sets the mode for cycle-accurate power analysis.

```
smartpower_set_mode_for_analysis -mode {value}
```

Arguments

`-mode {value}`

Specifies the mode for cycle-accurate power analysis.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example sets the mode for analysis to active:

```
smartpower_set_mode_for_analysis -mode {active}
```

smartpower_set_operating_condition

Sets the operating conditions used in SmartPower to one of the pre-defined types.

```
smartpower_set_operating_condition -opcond {value}
```

Arguments

`-opcond {value}`

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
best	Sets the operating conditions to best
typical	Sets the operating conditions to typical
worst	Sets the operating conditions to worst

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the operating conditions to best:

```
smartpower_set_operating_condition -opcond {best}
```

smartpower_set_pin_enable_rate

Enables you to annotate the enable rate value of a pin driving an enable pin. For I/Os, if you do not use this command, the enable rate of the IOEnableSet is used. For memories, if you do not use this command, the enable rate of the MemoriesEnableSet is used.

```
smartpower_set_pin_enable_rate -pin_name {pin_name} -pin_enable_rate {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of a pin for which the enable rate will be annotated. This pin must be the direct driver of an enable pin.

`-pin_enable_rate {value}`

Specifies the value of the pin enable rate as a percentage, which can be any positive decimal between 0 and 100, inclusive.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example sets the enable rate of the pin driving the enable pin of a bidirectional I/O

```
smartpower_set_pin_enable_rate -pin_name mybibuf/U0/U1:EOUT   \
-pin_enable_rate 50.4
```

See Also

smartpower_set_default_enable_rate

smart_power_remove_pin_enable_rate

Tcl documentation conventions

smartpower_set_pin_frequency

Sets the frequency of a pin in megahertz (MHz). If you do not use this command, each pin will have default frequency based on its domain.

```
smartpower_set_pin_frequency -pin_name {pin_name} -pin_freq {value}
```

Arguments

`-pin_name {pin_name}`

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

`-pin_freq {value}`

Specifies the value of the frequency in MHz, which can be any positive decimal number.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

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

Examples

This example sets the frequency of the pin named "count8_clock" to 100 MHz:

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

See Also

smartpower_remove_pin_frequency

smartpower_set_preferences

Sets the following preferences: power unit, frequency unit, operating mode, operating conditions, and toggle. These preferences can also be set from the preferences dialog box.

```
smartpower_set_preferences -powerunit {value} -frequnit {value} -opmode {value} -opcond
{value} -toggle {value}
```

Arguments

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
active	The operating mode is set to active
static	The operating mode is set to static
sleep	The operating mode is set to sleep
Flash*Freeze	The operating mode is set to Flash*Freeze
shutdown	The operating mode is set to shutdown

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- The following arguments have been removed. Running the script will trigger a warning message: Warning: Invalid argument: -argname "argvalue" Ignored. Ignore the warning.

-maxblocks {integer > 0}
-maxpins [{integer > 0}
-sortorder {ascending, descending}
-sortby {powervalues, alphabetical}

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best operating conditions are available only for certain families and devices.

Exceptions

- None

Examples

This example sets the frequency of the power unit to "watts", the frequency unit to "Hz", the operating mode to "active", the operating condition to "typical", and the toggle to "true":

```
smartpower_set_setpreferences -powerunit {w} -frequnit {hz} -opmode {active} -opcond
{typical} -toggle {true}
```

See Also

SmartPower Preferences

smartpower_set_scenario_for_analysis

Sets the scenario for cycle-accurate power analysis.

```
smartpower_set_scenario_for_analysis -scenario{value}
```

Arguments

`-scenario {value}`

Specifies the mode for cycle-accurate power analysis.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

The following example sets the scenario for analysis to my_scenario:

```
smartpower_set_scenario_for_analysis -scenario {my_scenario}
```

smartpower_set_temperature_opcond

Sets the temperature in the operating conditions to one of the pre-defined types.

```
smartpower_set_temperature_opcond -use{value}
```

Arguments

`-use{value}`

Specifies the temperature in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the temperature in the operating conditions as specified in the Device Selection Wizard in Designer
design	Sets the temperature in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the temperature in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the temperature in the operating conditions as specified in the custom mode-settings:

```
smartpower_set_temperature_opcond -use{mode}
```

smartpower_set_thermalmode

Sets the mode of computing junction temperature.

```
smartpower_set_thermalmode -mode {value}
```

Arguments

`-mode {value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- To compute the junction temperature, set the following three commands: smartpower_set_thermalmode, smartpower_set_tambient and smartpower_set_cooling. The junction temperature will be updated when an output command is executed, such as report(Power).

Exceptions

- None

Examples

The following example sets the computing of the junction temperature to ambient mode:

```
smartpower_set_thermalmode -mode {ambient}
```

smartpower_set_voltage_opcond

Sets the voltage in the operating conditions.

```
smartpower_set_voltage_opcond -voltage{value} -use{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-use{value}`

Specifies the voltage in the operating conditions for each voltage supply. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the voltage in the operating conditions as specified in the Device Selection Wizard in Designer
design	Sets the voltage in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the voltage in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the VCCA as specified in the SmartPower mode-specific settings:

```
smartpower_set_voltage_opcond -voltage{vcca} -use{mode}
```

smartpower_temperature_opcond_set_design_wide

Sets the temperature for SmartPower design-wide operating conditions.

```
smartpower_temperature_opcond_set_design_wide -best{value} -typical{value} -worst{value} -
thermal_mode{value}
```

Arguments

`-best{value}`

Specifies the best temperature (in degrees Celsius) used for design-wide operating conditions.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) used for design-wide operating conditions.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) used for design-wide operating conditions.

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the temperature for design-wide operating conditions to Best 20, Typical 30, and Worst 60:

```
smartpower_temperature_opcond_set_design_wide -best{20} -typical{30} -worst{60}
```

smartpower_temperature_opcond_set_mode_specific

Sets the temperature for SmartPower mode-specific operating conditions.

```
smartpower_temperature_opcond_set_mode_specific -mode{value} -best{value} -typical{value} -worst{value} -thermal_mode{value}
```

Arguments

`-mode{value}`

Selects the mode to which apply the operating condition settings. You can select a pre-defined mode or any custom mode in your design.

`-best{value}`

Specifies the best temperature (in degrees Celsius) for the selected mode.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) for the selected mode.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) for the selected mode.

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the temperature for mode-specific operating conditions for mode1:

```
smartpower_temperature_opcond_set_mode_specific -mode{mode1} -best{20} -typical{30} -
worst{60}
```

smartpower_voltage_opcond_set_design_wide

Sets the voltage settings for SmartPower design-wide operating conditions.

```
smartpower_voltage_opcond_set_design_wide -voltage{value} -best{value} -typical{value} -
worst{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for design-wide operating conditions.

`-typical{value}`

Specifies the typical voltage used for design-wide operating conditions.

`-worst{value}`

Specifies the worst voltage used for design-wide operating conditions.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets VCCA for design-wide to best 20, typical 30 and worst 40:

```
smartpower_voltage_opcond_set_design_wide -voltage{VCCA} -best{20} -typical{30} -
worst{40}
```

smartpower_voltage_opcond_set_mode_specific

Sets the voltage settings for SmartPower mode-specific use operating conditions.

```
smartpower_voltage_opcond_set_mode_specific -opmode{value} -voltage{value} -best{value} -
typical{value} -worst{value}
```

Arguments

`-opmode{value}`

Selects the mode to which apply the operating condition settings. You can select a pre-defined mode or any custom mode in your design.

`-voltage{value}`

Specifies the voltage in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for mode-specific operating conditions.

`-typical{value}`

Specifies the typical voltage used for mode-specific operating conditions.

-worst{*value*}

Specifies the worst voltage used for mode-specific operating conditions.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Notes

- None

Exceptions

- None

Examples

This example sets the voltage for the static mode and sets best to 20, typical to 30 and worst to 40:

```
smartpower_voltage_opcond_set_mode_specific -opmode{active} -voltage{VCCA} -best{20} -
typical{30} -worst{40}
```

st_commit

Saves the changes made in SmartTime to the design (.adb) file

```
st_commit
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

```
st_commit
```

See Also

st_restore

Tcl documentation conventions

st_create_set

Creates a set of paths to be analyzed. Use the arguments to specify which paths to include. To create a set that is a subset of a clock domain, specify it with `-clock` and `-type`. To create a set that is a subset of an inter-clock domain set, specify it with `-source_clock` and `-sink_clock`. To create a set that is a subset (filter) of an existing named set, specify the set to be filtered with `-from_set`.

To create a set that is not derived from an existing set, you must provide both the `-source` *`pin_list`* and `-sink`*`pin_list`* derived. Otherwise, the `-source` and `-sink` arguments act as filters on the pins from the parent set. You must give each new set a unique name in the design.

```
st_create_set -name name
[-parent_set name ]
[-clockclock_id -type value ]
[-in_to_out]
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list ] -sink pin_list ]
```

Arguments

`-name` *`name`*

Specifies a unique name for the newly create path set.

-parent_set *`name`*

Specifies the name of the set to filter.

-clock *`clock_id`*

Specifies that the set is to be a subset of the given clock domain. This argument is valid only if you also specify the -type argument.

`-type` *`value`*

Specifies the predefined set type on which to base the new path set. You can only use this argument with the -clock argument, not by itself.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

-in_to_out

Specifies that the set is based on the "Input to Output" set, which includes paths that start at input ports and end at output ports.

-source_clock *clock_id*

Specifies that the set will be a subset of an inter-clock domain set with the given source clock.

You can only use this option with the -sink_clock option, not by itself.

-sink_clock *clock_id*

Specifies that the set will be a subset of an inter-clock domain set with the given sink clock.

You can only use this option with the -source_clock option, not by itself.

-source *pin_list*

Specifies a filter on the source pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

-sink *pin_list*

Specifies a filter on the sink pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

```
st_create_set -name { my_user_set } -source { C* } -sink { D* }
st_create_set -name { my_other_user_set } -from_set { my_user_set } -source { CL* }
st_create_set -name { adder } -clock { ALU_CLOCK } -type { REG_TO_REG }  -sink { ADDER* }
st_create_set -name { another_set } -source_clock { EXTERN_CLOCK } -sink_clock { MY_GEN_CLOCK }
st_create_set -name { some_p2p } -pin2pin -to { T* }
```

See Also

Tcl documentation conventions

st_remove_set

st_edit_set

Modify the paths in a user set.

```
st_edit_set -name name
[-source pin_list ] [-sink  pin_list ]
[-rename_to name ]
```

Arguments

`-name name`

Specifies the name of the set to modify.

`-source pin_list`

If the set is a subset of another set, specifies a filter on the source pins from the parent set. Otherwise, this option specifies the source pins of the set.

`-sink pin_list`

If the set is a subset of another set, specifies a filter on the sink pins from the parent set. Otherwise, this option specifies the sink pins of the set.

`-rename_to name`

Specifies a new name for the set.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC^{PLUS}, ProASIC, and Axcelerator

Exceptions

None

Examples

```
st_edit_set -name { my_user_set} -rename_to { my_critical_pins }
st_edit_set -name { adder } -sink { ADD* }
```

See Also

Tcl documentation conventions

st_create_set

st_remove_set

st_expand_path

Displays expanded path information (path details) for paths. The paths to be expanded are identified by the parameters required to display these paths with st_list_paths. For example, to expand the first path listed with st_list_paths -clock {MYCLOCK} -type {register_to_register}, use the command st_expand_path -clock {MYCLOCK} -type {register_to_register}. Path details contain the pin name, type, net name, cell name, operation, delay, total delay, and edge as well as the arrival time, required time, and slack. These details are the same as details available in the SmartTime Expanded Path window.

```
st_expand_path [-set name]
[-clock clock_id -type value]
[-in_to_out]
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list] [-sink pin_list]
[-analysis value]
[-index list_of_indices]
[-format value]
```

Arguments

-set *name*

Displays a list of paths from the named set. You can either use the -set option to specify a set name, or use both -clock and -type to specify a set. A list of valid set names includes "in_to_out", as well as any user set names.

-clock *clock_id*

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set "Input to Output, which includes paths that start at input ports and end at output ports.

`-type` *value*

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` *clock_id*

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` *clock_id*

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` *pin_list*

Specifies a filter on the source pins of the paths to be listed.

`-sink` *pin_list*

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *name*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-index` *`list_of_indices`*

Specifies which paths to display. The index starts at 1 and defaults to 1. Only values lower than the max_paths option will be expanded.

`-format` *`value`*

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value fie format

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

Note: The following example returns a list of five paths:

```
st_expand_path -clock { myclock } -type {reg_to_reg }
st_expand_path -clock {myclock} -type {reg_to_reg} -index { 1 2 3 } -format text
```

See Also

Tcl documentation conventions

st_list_paths

st_list_paths

Displays the list of paths in the same tabular format shown in SmartTime.

```
st_list_paths [-set name ]
[-clock clock_id -type value ]
[-in_to_out]
[-source_clock clock_id -sink_clock  clock_id]
[-source pin_list ] [-sink pin_list ]
[-analysis value ]
[-format value ]
```

Arguments

`-set` *`name`*

Displays a list of paths from the named set. You can either use the –set option to specify a set name, or use both –clock and –type to specify a set. A list of valid set names includes “in_to_out”, as well as any user set names.

-clock *`clock_id`*

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set “Input to Output”, which includes paths that start at input ports and end at output ports.

`-type` *`value`*

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` **`pin_list`**

Specifies a filter on the source pins of the paths to be listed.

`-sink` **`pin_list`**

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *`name`*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-format value`

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value fie format

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

```
st_list_paths -set  { myset }
st_list_paths -analysis mindelay -clock { myclock } -type { reg_to_reg } -format csv
```

The list of paths can be written to a file with the following Tcl commands:

```
set outfile [ open "pathlisting.csv" w]
puts $outfile [ st_list_paths -format csv -set { myset} ]
close $outfile
```

See Also

Tcl documentation conventions

st_expand_path

st_remove_set

Deletes a user set from the design.

```
st_remove_set -name name
```

Arguments

`-name name`

Specifies the name of the set to delete.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, and Axcelerator

Exceptions

None

Examples

```
st_remove_set { clockset1 }
```

See Also

Tcl documentation conventions

st_create_set

st_restore

Restores constraints previously committed in SmartTime.

```
st_restore
```

Arguments

None

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A

Exceptions

None

Examples

```
st_restore
```

See Also

st_commit

Tcl documentation conventions

st_set_options

Sets options for timing analysis. With no parameters given, it will display the current settings of the options. For IGLOO, Fusion, ProASIC3, ProASIC[PLUS], and Axcelerator families, these options also affect timing-driven place-and-route.

```
st_set_options [-max_opcond value ]
[-min_opcond value]
[-interclockdomain_analysis value]
[-use_bibuf_loopbacks value]
[-enable_recovery_removal_checks value]
[-break_at_async value]
[-filter_when_slack_below value]
[-filter_when_slack_above value]
[-remove_slack_filters]
[-limit_max_paths value]
[-expand_clock_network value]
[-expand_parallel_paths value]
[-analysis_scenario value]
[-tdpr_scenario value]
[-reset]
```

Arguments

-max_opcond *value*

Sets the operating condition to use for Maximum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
worst	Use Worst Case conditions for Maximum Delay Analysis
typ	Use Typical conditions for Maximum Delay Analysis
best	Use Best Case conditions for Maximum Delay Analysis

-min_opcond *value*

Sets the operating condition to use for Minimum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
best	Use Best Case conditions for Minimum Delay Analysis
typ	Use Typical conditions for Minimum Delay Analysis
worst	Use Worst Case conditions for Minimum Delay Analysis

-interclockdomain_analysis *value*

Enables or disables inter-clock domain analysis.

Value	Description
yes	Enables inter-clock domain analysis
no	Disables inter-clock domain analysis

-use_bibuf_loopbacks *value*

Enables or disables loopback in bibufs.

Value	Description
yes	Enables loopback in bibufs
no	Disables loopback in bibufs

-enable_recovery_removal_checks *value*

Enables or disables recovery and removal checks.

Value	Description
yes	Enables recovery and removal checks
no	Disables recovery and removal checks

-break_at_async *value*

Enables or disables breaking paths at asynchronous ports.

Value	Description
yes	Enables breaking paths at asynchronous ports
no	Disables breaking paths at asynchronous ports

-filter_when_slack_below *value*

Do not show paths with slack below x.

-filter_when_slack_above *value*

Do not show paths with slack above y.

-remove_slack_filters

Remove all existing slack filters.

-limit_max_paths *value*

Limit path reporting commands to a maximum of <n> paths, where n is a value of 0 or higher.

-expand_clock_network *value*

Enables or disables expanded clock network information in expanded paths.

Value	Description
yes	Enables expanded clock network information in paths
no	Disables expanded clock network information in paths

-expand_parallel_paths *value*

Expand a maximum of <n> parallel paths, where n is a value of 0 or higher. If n is 0 or 1, only one path will be expanded when viewing expanded paths.

-analysis_scenario *value*

Set the timing constraints scenario to be used for both maximum delay and minimum delay analysis. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for TDPR.

-tdpr_scenario *value*

Set the timing constraints scenario to be used by the place and route engine. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for analysis.

-reset

Reset all options to their default values, except for scenarios used for analysis and TDPR that remain unchanged.

Supported Families

IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A

Exceptions

None

Examples

```
st_set_options -max_opcond worst \
-min_opcond best \
-interclockdomain_analysis true \
-enable_removal_recovery_checks true
st_set_options -limit_max_paths 50 -remove_slack_filters \
-filter_when_slack_above 3
```

See Also

Tcl documentation conventions

timer_add_clock_exception

Adds an exception to or from a pin with respect to a specified clock.

```
timer_add_clock_exception -clock clock_name -pin pin_name -dir value
```

Arguments

-clock span class="tcl_value"clock_name

Specifies the name of the clock.

-pin *pin_name*

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

-dir *value*

Specifies direction.

Value	Description
from	Refers to paths starting at the specific pin.
to	Refers to paths ending at the specific pin.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-Afamilies, the timing analysis tool translates this command to an equivalent SDC command set_muliticycle_path.

Examples

The following example adds 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}
```

The following example adds 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}
```

See Also

timer_remove_clock_exception

set_multicycle_path

Tcl documentation conventions

timer_add_pass

Adds a pin to the list of pins that the path must be shown passing through in the timing analysis tool.

```
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.

Supported Families

All

Description

When you set a pass on a module pin, you can see a path through individual pins.

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, the timing analysis tool ignores the timer_add_pass command.

Examples

This example adds a pass through the pin "reg_q_a_0_:CLK":

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

This example adds a pass through a clear pin "reg_q_a_0_:CLR":

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

See Also

timer_add_stop

Tcl documentation conventions

timer_add_stop

Adds the specified pin to the list of pins through which the paths will not be displayed in the timing analysis tool.

```
timer_add_stop -pin pin_name
```

Arguments

`-pin pin_name`

Specifies the name of the pin through which the path will not be displayed.

Supported Families

All

Description

Without stop points, you see all paths from pad to pad in the design. If you do not want to see the paths going through register clock pins, you can specify these as stop points. The path going through the specified pins will not be displayed.

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, Actel recommends that you use the equivalent SDC command set_false_path.

Examples

The following example adds a stop to the pin "a<2>":

```
timer_add_stop -pin {a<2>}
```

The following example adds a stop to a clock and the clear pin "reg_q_a_0_:CLR":

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

See Also

timer_add_pass

set_false_path (SDC false path constraint)

Tcl documentation conventions

timer_commit

Saves the changes made to constraints into the Designer database.

```
timer_commit
```

Arguments

None

Supported Families

All

Exceptions

- For theIGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, Actel recommends that you use import_source . This automatically commits constraints to the design database.

Examples

```
timer_commit
```

See Also

import_source

timer_restore

Tcl documentation conventions

timer_get_path

Displays the path between the specified pins in the Log window.

```
timer_get_path -from source_pin -to destination_pin
[-exp value]\
[-sort value]\
[-order value]\
[-case value]\
[-maxpath maximum_paths]\
[-maxexpath maximum_paths_to_expand]\
[-mindelay minimum_delay]\
[-maxdelay maximum_delay]\
[-breakatclk value]\
[-breakatclr value]
```

Arguments

`-from source_pin`

Specifies the name of the source pin for the path.

`-to destination_pin`

Specifies the name of the destination pin for the path.

`-exp value`

Specifies whether to expand the path. The following table shows the acceptable values for this argument:

Value	Description
yes	Expands the path
no	Does not expand the path

`-sort value`

Specifies whether to sort the path by either the actual delay or slack value. The following table shows the acceptable values for this argument:

Value	Description
actual	Sorts the path by the actual delay value
slack	Sorts the path by the slack value

`-order value`

Specifies whether the maximum list size is based on the longest or shortest paths. The following table shows the acceptable values for this argument:

Value	Description
long	Base the maximum list size on the longest path in the design
short	Base the maximum list size on the shortest path in the design

`-case value`

Specifies whether the report will include timing values for the worst, typical, or best cases. The following table shows the acceptable values for this argument:

Value	Description
worst	Includes timing values for the worst cases
typ	Includes timing values for typical cases
best	Includes timing values for the best cases

`-maxpath maximum_paths`

Specifies the maximum number of paths to display.

`-maxexpath maximum_paths_to_expand`

Specifies the maximum number of paths to expand.

`-mindelay minimum_delay`

Specifies the path delay in the minimum delay analysis mode.

`-maxdelay maximum_delay`

Specifies the path delay in the maximum delay analysis mode.

`-breakatclk value`

Specifies whether to break the paths at the register clock pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clock pins
no	Does not break the paths at the register clock pins

`-breakatclr` *value*

Specifies whether to break the paths at the register clear pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clear pins
no	Does not break the paths at the register clear pins

Supported Families

All

Exceptions

None

Examples

The following example returns the paths from input port headdr_dat<31> to the input pin of register u0_headdr_data1_reg/data_out_t_31 under typical conditions.

```
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"
```

The following example returns the paths from the clock pin of register gearbox_inst/bits64_out_reg<55> to the output port pma_tx_data_64bit[55]

```
timer_get_path -from "gearbox_inst/bits64_out_reg<55>/U0:CLK" \
     -to {pma_tx_data_64bit[55]} \
     -exp "yes"
```

See Also

Tcl documentation conventions

timer_get_clock_actuals

Displays the actual clock frequency in the Log window, when the timing analysis tool is initiated.

```
timer_get_clock_actuals -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the frequency (or period) to display.

Supported Families

All

Exceptions

None

Examples

This example displays the actual clock frequency of clock clk1 in the Log window:

```
timer_get_clock_actuals -clock clk1
```

See Also

timer_get_clock_constraints

Tcl documentation conventions

timer_get_clock_constraints

Returns the constraints (period, frequency, and duty cycle) on the specified clock.

```
timer_get_clock_constraints -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the constraint to display.

Supported Families

All

Exceptions

None

Examples

The following example displays the clock constraints on the clock clk in the Log window:

```
timer_get_clock_constraints -clock clk
```

See Also

timer_get_clock_actuals

Tcl documentation conventions

timer_get_maxdelay

Displays the maximum delay constraint between two pins in a path in the Log window.

```
timer_get_maxdelay -from source_pin -to destination_pin
```

Arguments

`-from source_pin`

Specifies the name of the source pin in the path.

`-to destination_pin`

Specifies the name of the destination pin in the path.

Supported Families

All

Exceptions

None

Examples

The following example displays the maximum delay constraint from the pin clk166 to the pin reg_q_a_9_/U0:CLK in the Log window:

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

See Also

timer_set_maxdelay

Tcl documentation conventions

timer_get_path_constraints

Displays the path constraints that were set as the maximum delay constraint in the timing analysis tool.

```
timer_get_path_constraints
```

Arguments

None

Supported Families

All

Description

This command lists the paths constrained by maximum delay values. The information is displayed in the Log window.

If no maximum delay constraints are set, this command does not report anything.

Exceptions

None

Examples

Invoking timer_get_path_constraints displays the following paths and their delay constraints in the Log window:

```
max_delay -from [all_inputs] -to [all_outputs] = 12 ns
max_delay -from p_f_testwdata0 p_f_testwdata1 -to p_f_dacuwdata0 p_f_dacuwdata1
r_f_dacuwdata0 r_f_dacuwdata1 = 8 ns
```

See Also

timer_set_maxdelay

Tcl documentation conventions

timer_remove_clock_exception

Removes the previously set clock constraint.

```
timer_remove_clock_exception -clock clock_name -pin pin_name -dir value
```

Arguments

`-clock` ***clock_name***

Specifies the name of the clock from which to remove the constraint.

`-pin` ***pin_name***

Specifies the name of the pin to remove.

`-dir` ***value***

Specifies direction.

Value	Description
from	Refers to paths starting at the specified pin.
to	Refers to paths ending at the specified pin.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, the timing analysis tool ignores the timer_remove_clock_exceptions command. To remove the clock exception, use the Set Multicycle Constraint dialog box.

Examples

This example removes 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}
```

This example removes 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

Tcl documentation conventions

Set Multicycle Constraint dialog box

timer_remove_pass

Removes the previously entered path pass constraint.

```
timer_remove_pass -pin pin_name
```

Arguments

`-pin pin_name`

Specifies the name of the pin from which to remove the path pass constraint.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, the timing analysis tool ignores the timer_remove_pass command.

Examples

The following example 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

Tcl documentation conventions

timer_remove_stop

Removes the previously entered path stop constraint on the specified pin.

```
timer_remove_stop -pin pin_name
```

Arguments

`-pin pin_name`

Specifies the name of the pin from which to remove the path stop constraint.

Supported Families

All

Description

If you remove a path stop constraint using the Timer GUI, and then export a script using **File** > **Export** > **Script files**, the resulting script will contain `timer_remove_pass -pin pin_name` instead of `timer_remove_stop -pin pin_name`.

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, eX, and SX-A families, Actel recommends the following flow:
    1. Open **SmartTime** > Set False Path Constraint dialog box.
    2. Look for the pin name in the **Through:** list (You must not have any entry selected in the **From** or **To** lists).
    3. Delete this pin.

Examples

The following example 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

Tcl documentation conventions

Set False Path Constraint dialog box

timer_restore

Restores constraints previously committed in Timer.

```
timer_restore
```

Arguments

None

Supported Families

All

Exceptions

None

Examples

```
timer_restore
```

See Also

timer_commit

Tcl documentation conventions

timer_setenv_clock_freq

Sets a required clock frequency for the specified clock in megahertz (MHz).

```
timer_setenv_clock_freq -clock clock_name -freq value [-dutycycle dutycycle]
```

Arguments

-clock *clock_name*

Specifies the name of the clock for which to set the required frequency.

-freq *value*

Specifies the frequency in MHz.

-dutycycle *dutycycle*

Specifies the duty cycle for the clock constraint.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, SX-S, SX-A, and eX families, Actel recommends that you use the equivalent SDC command create_clock.

Examples

The following example sets a clock frequency of 100MHz on the clock clk1:

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

See Also

create_clock (SDC clock constraint)

Tcl documentation conventions

timer_setenv_clock_period

timer_setenv_clock_period

Sets the clock period constraint on the specified clock.

```
timer_setenv_clock_period -clock clock_name \
[-unit {value}] -period period_value\
[-dutycycle dutycycle]
```

Arguments

-clock *clock_name*

Specifies the name of the clock for which to set the period.

-unit {*value*}

Specifies the unit for the clock period constraint. The default is ns. The following table shows the acceptable values for this argument:

Value	Description
Ns	nanoseconds
Ps	picoseconds

`-period` ***`period_value`***

Specifies the period in the specified unit.

`-dutycycle` ***`dutycycle`***

Specifies the duty cycle for the clock constraint.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, SX-S, SX-A, and eX families, Actel recommends that you use the equivalent SDC command create_clock.

Examples

The following example sets a clock period of 2.40ns on the clock clk1:

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

See Also

timer_setenv_clock_freq

Tcl documentation conventions

timer_set_maxdelay

Adds a maximum delay constraint to the specified path.

```
timer_set_maxdelay -from source_pin \
-to destination_pin \
[-unit {value}] -delay delay_value
```

Arguments

`-from` ***`source_pin`***

Specifies the name of the source pin in the path.

`-to` ***`destination_pin`***

Specifies the name of the destination pin in the path.

`-unit {`***`value`***`}`

Specifies whether the delay unit is in nanoseconds or picoseconds. The following table shows the acceptable values for this argument:

Value	Description
ns	Sets the delay in nanoseconds
ps	Sets the delay in picoseconds

`-delay` *`delay_value`*

Specifies the actual delay value for the path.

Supported Families

All

Exceptions

- For the IGLOO, Fusion, ProASIC3, ProASIC[PLUS], ProASIC, Axcelerator, SX-S, SX-A, and eX families, Actel recommends that you use the equivalent SDC command set_max_delay.

Examples

The following example sets a maximum delay of 20 nanoseconds from register reg1 to output pin out1:

```
timer_set_maxdelay -from {reg1:CLK} -to {out1} -unit {ns} -delay 20.00
```

See Also

timer_get_maxdelay

set_max_delay (SDC max path constraint)

Tcl documentation conventions

timer_remove_all_constraints

Removes all timing constraints in the current design.

```
timer_remove_all_constraints
```

Arguments

None

Supported Families

All

Exceptions

None

Examples

The following example removes all of the constraints from the design and then commits the changes:

```
timer_remove_all_constraints
timer_commit
```

See Also

timer_commit

Tcl documentation conventions

use_file

Specifies which file in your project to use.

```
use_file
-file value
-module value
-designer_view value
```

Arguments

-file*value*

Specifies the EDIF or ADB file you wish to use in the project. Value is the name of the file you wish use (including the full pathname).

-module *value*

Specifies the module in which you want to use the file.

-designer_view *value*

Specifies the Designer View in which you wish to use the file.

Supported Families

All

Exceptions

- None

Example

Specify file1.edn in the ./project/synthesis directory, in the module named top, in the Designer View named impl1.

```
use_file -file "./project/synthesis/file1.edn" -module "top" -designer_view "Impl1"
```

See Also

use_source_file

use_source_file

Defines a module for your project.

```
use_source_file
-file value
-module value
```

Arguments

`-file` *value*

Specifies the Verilog or VHDL file. Value is the name of the file you wish use (including the full pathname).

`-module` *value*

Specifies the module in which you want to use the file.

Supported Families

All

Exceptions

- None

Example

Specify file1.vhd in the ./project/hdl directory, in the module named top.

```
use_source_file -file "./project/hdl/file1.vhd" -module "top"
```

See Also

use_file

Product Support

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

Customer Service

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

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

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

Actel Technical Support

Visit the Actel Customer Support website (http://www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at http://www.actel.com/.

Contacting the Customer Technical Support Center

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

Email

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

The technical support email address is tech@actel.com.

Phone

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

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

For more information about Actel's products, visit our website at

www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 • USA*
Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom*
Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540

Actel Japan *• EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • http://jp.actel.com

Actel Hong Kong *• Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

5-02-9122-22/7.08