Actel® Libero™ v2.3

User´s Guide

Windows®

Actel Corporation, Sunnyvale, CA 94086

Printed in the United States of America

Part Number: 5029129-3

Release: December 2002

Trademarks

Table of Contents

List of Figures

Introduction

Libero™ is Actel's integrated design environment (IDE). Libero IDE integrates your design tools, streamlines the design flow, manages all design and log files, and passes design data between tools.

Libero IDE Installation and Licensing

For information on installation and licensing see the *Actel Installation Guide.*The *Installation and Licensing FAQ* is also available from the Start menu. From Start, choose Programs, Libero 2.3, and click *Installation and Licensing FAQ*.

Libero IDE Editions

Libero IDE is offered in Silver, Gold, and Platinum editions. Refer to online help for more on the differences between editions.

Check the license.dat file to determine which package you are using.

Table A-1. Libero Editions

Libero Edition	**License Feature**
Libero Evaluation	ACTEL_EVAL
Libero Silver	ACTEL_BASE
Libero Gold	ACTEL_VISTA
Libero Platinum	ACTEL_SUMMIT

Libero Tools

Libero IDE combines tools from Actel and EDA powerhouses to offer you a one stop FPGA design solution.

Table 1-2. Libero Tools

Function	Tool	Company
Design Entry (HDL)	HDL Editor	Actel
Design Entry (Schematic)	ViewDraw™	Actel
Synthesis	Synplify® or Synplify® Lite	Synplicity®
Simulation	ModelSim®	Mentor Graphics®
Automatic Test Bench Generator	WaveFormer Lite™	SynaptiCAD™
Automatic Macro Generator	ACTgen	Actel
Place-and-route	Designer Series	Actel
Programming Software	APS/Silicon Sculptor	Actel
In-Silicon Debug	Silicon Explorer II	Actel

Document Organization

This guide provides detailed information about Libero. Updates to this User's Guide are posted at http://www.actel.com/techdocs/manuals/index.html.

This guide is divided into the following chapters:

Chapter 1 "Libero IDE Design Flows" on page 21 describes schematic, HDL, and mixed flows.

Chapter 2 "Getting Started with Libero IDE" on page 33 describes Libero user interface and commands.

Chapter 3 "Managing Your Project" on page 47 contains information on creating and managing projects using Libero.

Chapter 4 "Design Entry - HDL Sources" on page 57 contains information on the Libero HDL Editor.

Chapter 5 "Design Entry - Schematic Sources" on page 63 contains information on how to create schematics sources for your design.

Chapter 6 "Synthesis" on page 105 contains details about running synthesis using Synplify.

Chapter 7 "Design Implementation" on page 113 contains information on using Designer to implement your design.

Chapter 8 "Creating a Test Bench" on page 157 contains information on creating your test bench using WaveFormer Lite from SynaptiCAD.

Chapter 9 "Simulation" on page 177 contains information on performing functional/behavioral, structural, and timing simulations using Model*Sim* for Actel from Model Technology.

Chapter 10 "Programming" on page 213 contains information about creating your programming file for Actel device.

Appendix A "Special Features" on page 221 contains information on features that are not part of the normal design flow.

Appendix B "Product Support" on page 225 contains information on Actel Product Support.

Document Assumptions

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

1. You have installed Libero IDE.
2. You are familiar with FPGA architecture and FPGA design software.

Platform Support

Libero is designed for use on PCs running Windows 98, NT 4.0, Windows 2000, or Windows XP.

Your Comments

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

Actel Manuals

Libero IDE includes printed and online manuals. The online manuals are in PDF format and available from the Libero Start Menu and on the CD-ROM.

From the Start menu choose:

- Programs > Libero 2.3 > Libero 2.3 Documentation.
- Programs > Designer Series > R2-2002 Documentation

From the CD, insert your CD-ROM and click *Documentation* from the main screen, or look on the CD-ROM in the "/doc" directory. These manuals are also installed onto your system when you install the Libero software. To view the online manuals, you must install Adobe® Acrobat Reader® from the CD-ROM disc.

Online Help

Libero comes with online help. Online help specific to each Actel software tool is available for Designer, ACTgen, Silicon Expert, Silicon Explorer II, and Silicon Sculptor.

Also, our integrated tool suite from ViewDraw, Model*Sim*, Synplify, and WaveFormer Lite, all come with online help.

1

Libero IDE Design Flows

This chapter contains information on the three Libero design flows:

- "Schematic Based Design Flow" on page 21
- "HDL Synthesis Based Design Flow" on page 25
- "Mixed Schematic-HDL Design Flow Overview" on page 29

Each flow is a multi-step, iterative process that includes creation, implementation, programming and system verification.

Schematic Based Design Flow

The Libero schematic based design flow has four main steps:

1. Design Creation
2. Design Implementation
3. Device Programming
4. System Verification

These steps are illustrated in Figure 1-1 and described in detail in the sections which follow.

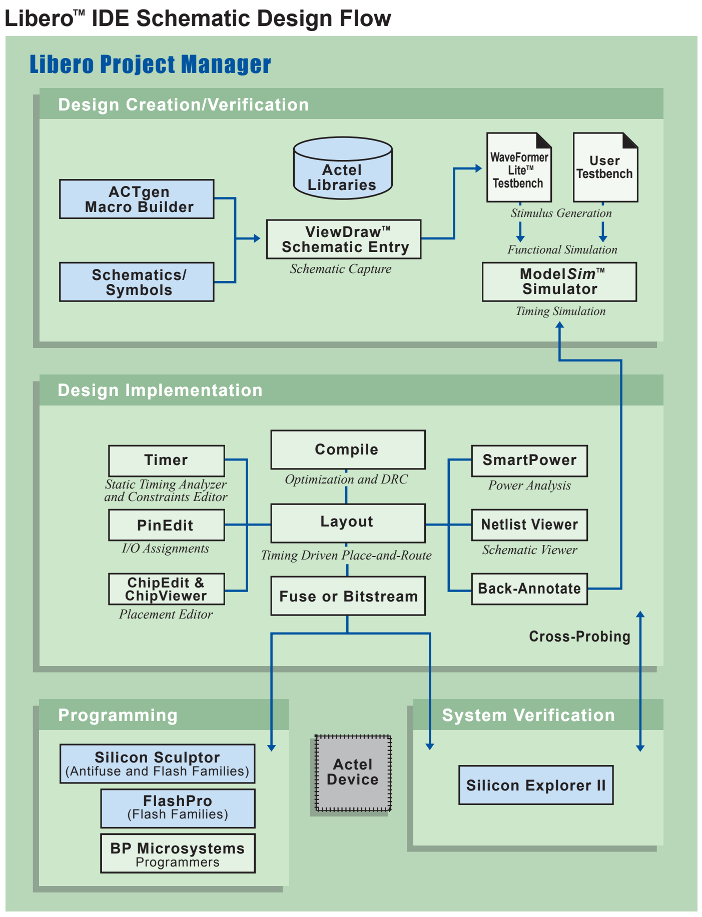


Figure 1-1. Libero IDE Schematic Design Flow

Design Creation

Design creation consists of capturing a schematic representation of the design and performing functional simulations with a test bench.

Design Capture

For schematic capture, Libero uses ViewDraw for Actel, which includes a schematic editor. The schematic editor provides a graphical entry method to capture designs. Refer to"Design Entry - Schematic Sources" on page 63 for information about design capture using ViewDraw for Actel.

Your structural HDL netlist is automatically created after you use the *Save + Check* command in ViewDraw.

Adding ACTgen Macros

After using ACTgen to create HDL macros, use Libero to create symbols for the macros. Then using ViewDraw for Actel, instantiate the symbols into your schematic. Refer to "Adding ACTgen Macros" on page 65.

Creating a Test Bench

You must create a test bench and associate it with your project in order to run simulation. Use the HDL editor or SynaptiCAD's WaveFormer Lite to create your test bench. Refer to "Creating a Test Bench" on page 157 for information creating test benches.

Functional Simulation

Functional simulation verifies that the logic of a design is functionally correct. Refer to "Simulation" on page 177 for information about performing functional simulation using the Libero integrated simulator, Model*Sim* for Actel.

Design Implementation

During design implementation, Designer places-and-routes your design. You can also use Designer's User's tools, which are described in Table A-1.

Table A-1. Designer User Tools

User Tool	Function
Timer	Static timing analysis
SmartPower	Power analysis
ChipEdit	Customize I/O and logic macro placements
PinEdit	Customize I/O placements and attributes
Netlist Viewer	View your netlist and trace paths

After place-and-route, use ModelSim for Actel for timing simulation.

Place-and-Route

Start Designer from Libero to place-and-route your design. Refer to "Design Implementation" on page 113 and the *Designer User's Guide* for information about using Designer.

Static Timing Analysis

Use Designer's Timer tool to perform static timing analysis on your design. Refer to the *Timer User's Guide* for information about using Timer.

Creating a Test Bench

If you did not perform funtional simulation, you must now create a test bench and associate it with your project in order to run timing simulation. Use the HDL editor or SynaptiCAD's WaveFormer Lite to create your test bench. Refer to "Creating a Test Bench" on page 157 for information creating test benches.

Timing Simulation

Perform a timing simulation on your design after place-and-route in Designer. Timing simulation requires information extracted and back-annotated from Designer. Refer to "Timing Simulation" on page 175 for information about performing timing simulation.

Programming

Program the device with programming software and hardware from Actel or a supported 3rd party programming system. Refer to "Programming" on page 213 for information about programming an Actel device.

System Verification

You may perform system verification on a programmed antifuse device using the Actel Silicon Explorer diagnostic tool. Refer to the *Silicon Explorer QuickStart* for information about using the Silicon Explorer.

HDL Synthesis Based Design Flow

The HDL synthesis-based design flow has four main steps:

1. Design Creation
2. Design Implementation
3. Programming
4. System Verification

These steps are illustrated in Figure 1-2 and described in detail in this section.

Libero™ IDE HDL Design Flow

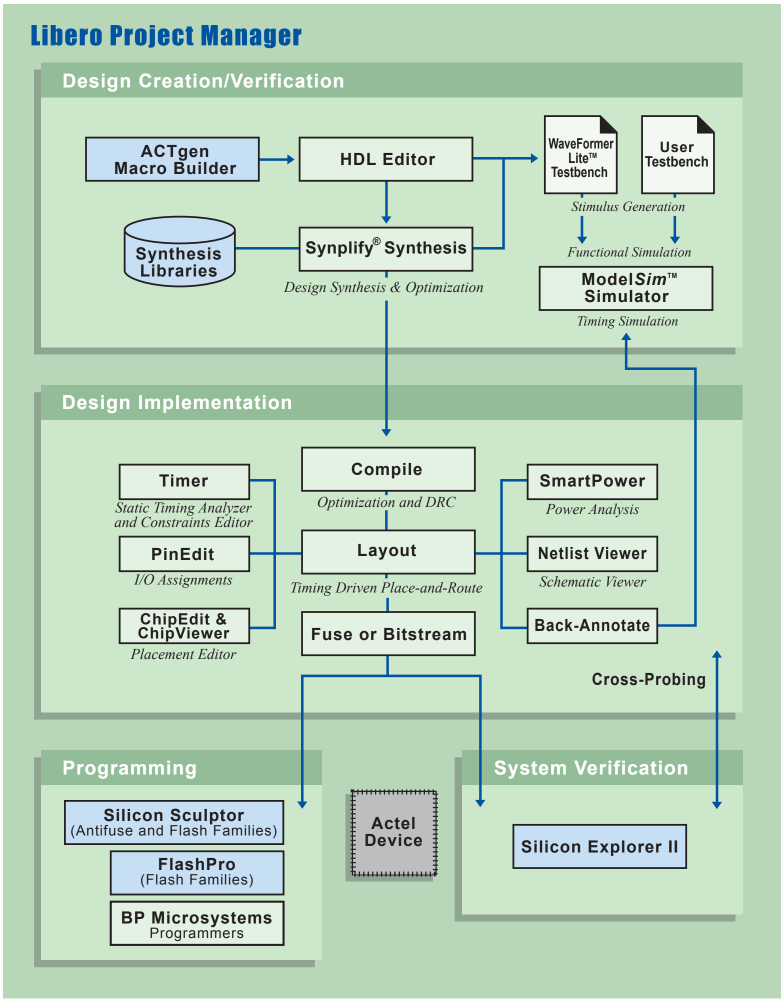


Figure 1-2. Libero IDE HDL Design Flow

Design Creation

Design Creation consists of using the HDL source editor to capture a behavioral description of a design, performing functional simulation with a test bench and running synthesis to produce an EDIF netlist.

Design Capture

During design entry, the Libero HDL Editor captures a design in an RTL-level (behavioral) source file. Your HDL design source may contain RTL-level constructs as well as instantiations of structural elements, such as ACTgen macros.

Adding ACTgen Macros

The ACTgen Macro Builder allows you to instantly create macros customized to your needs. After creating your ACTgen macro, instantiate the macro into your HDL code. For more information, refer to "Adding ACTgen Macros" on page 62.

Creating a Test Bench

You must create a test bench and associate it with your project in order to run timing simulation. Use the HDL editor or SynaptiCAD's WaveFormer Lite to create your test bench. Refer to "Creating a Test Bench" on page 157 for information on creating test benches.

Pre-Synthesis Simulation

Use functional simulation to verify that the logic of a design is functionally correct. Refer to "Functional Simulation" on page 173 for information about performing functional simulation using the Libero integrated simulator, ModelSim for Actel. Libero also includes SynaptiCAD's WaveFormer Lite, an automated test bench/test fixture creation tool. Refer to "Creating a Test Bench" on page 151 for information about using WaveFormer Lite.

Synthesis Netlist Generation

After you have entered your HDL source, you must synthesize it to generate a netlist. Synthesis transforms the behavioral HDL source into a gate-level netlist and optimizes the design for a target technology. Refer to "Synthesis" on page 105 for information about performing synthesis using the Libero integrated tool, Synplify.

Creating a Test Bench

If you did not perform funtional simulation, you must now create a test bench and associate it with your project in order to run timing simulation. Use the HDL editor or SynaptiCAD's WaveFormer Lite to create your test bench. Refer to "Creating a Test Bench" on page 157 for information about creating test benches.

Post-Synthesis Simulation

Use ModelSim for Actel to simulate your design before place-and-route. Post-synthesis simulation assists you in verifying the functionality of your post-synthesis structural HDL netlist.

Design Implementation

During design implementation, Actel's Designer places-and-routes your design. You can also use Designer's User's Tools, which are described in Table A-2.

Table A-2. Designer User Tools

User Tool	Function
Timer	static timing analysis
SmartPower	power analysis
ChipEdit	customize I/O and logic macro placements
PinEdit	customize I/O placements and attributes
Netlist Viewe	view your netlist and trace paths

After place-and-route, perform post-layout (timing) simulation.

Place-and-Route

Start Designer from Libero to place-and-route your design. Refer to "Design Implementation" on page 113 and the *Designer User's Guide* for information about using Designer.

Timing Simulation

Perform timing simulation on your design after place-and-route in Designer. Timing simulation requires information extracted and back-annotated from Designer. Refer to "Timing Simulation" on page 175 for information about performing timing simulation.

Programming

Program the device with programming software and hardware from Actel or a supported 3rd party programming system. Refer to "Programming" on page 213 for information about programming an Actel device.

System Verification

You may perform system verification on a programmed antifuse device using the Actel Silicon Explorer diagnostic tool. Refer to the *Silicon Explorer QuickStart* for information about using the Silicon Explorer.

Mixed Schematic-HDL Design Flow Overview

The Libero mixed schematic-HDL synthesis-based design flow has four main steps:

1. Design Creation
2. Design Implementation
3. Programming
4. System Verification

These steps are described in the following sections.

Design Creation

During design creation, ViewDraw for Actel captures a schematic representation of your design, the HDL Editor captures a behavioral description of your design, ModelSim performs a functional simulation with or without a test bench/test fixture, and synthesis produces an EDIF netlist.

Design Capture - Schematic

For schematic capture, Libero starts ViewDraw for Actel. The schematic editor provides a graphical entry method to capture designs. Refer to "Design Entry -

Schematic Sources" on page 63 for information about design capture using ViewDraw for Actel.

Design Capture - HDL

During design entry, the Libero HDL Editor captures a design in an RTL-level (behavioral) source file. Your HDL design source may contain RTL-level constructs as well as instantiations of structural elements, such as ACTgen macros. For mixed design flow, the top level must be a schematic with instantiated HDL and schematic blocks.

Adding ACTgen Macros

The ACTgen Macro Builder allows you to instantly create macros customized to your needs. These macros can them be added to your project by using ViewDraw or the HDL Editor, refer to page 62 and page 65.

Pre-Synthesis Simulation

Functional simulation verifies that the logic of a design is functionally correct. Refer to "Functional Simulation" on page 173 for information about performing functional simulation using the Libero integrated simulator, ModelSim for Actel. Libero also includes SynaptiCAD's WaveFormer Lite, an automated test bench/test fixture creation tool. Refer to "Creating a Test Bench" on page 151 for information about using WaveFormer Lite.

Synthesis & Netlist Generation

After you have entered your mixed design source, you must synthesize it to generate a netlist. Synthesis transforms the behavioral HDL source into a gate-level netlist and optimizes the design for a target technology. Refer to "Synthesis" on page 105 for information about performing synthesis.

If the HDL blocks in your schematic are only composed of ACTgen macros (no RTL-level HDL source files or state diagrams), you can skip the synthesis step and run Designer.

Structural (Post-Synthesis) Simulation

Structural simulation verifies the functionality of your post-synthesis structural HDL netlist.

Design Implementation

During design implementation, Designer places-and-routes your design. You can also use Designer's Timer tool to perform static timing analysis, ChipEdit to view and manually place I/O and logic macros, and PinEdit to customize I/O placements and attributes. After place-and-route, Libero performs post-layout (timing) simulation.

Place-and-Route

Start Designer from Libero to place-and-route your design. Refer to "Design Implementation" on page 113 and the *Designer User's Guide* for information about using Designer.

Static Timing Analysis

Use Designer's Timer tool to perform static timing analysis on your design. Refer to the *Timer User's Guide* for information about using Timer.

Timing Simulation

Perform a timing simulation on your design after place-and-route in Designer. Timing simulation requires information extracted and back-annotated from Designer. Refer to "Timing Simulation" on page 175 for information about performing timing simulation.

Programming

Program the device with programming software and hardware from Actel or a supported 3rd party programming system. Refer to "Programming" on page 213 for information about programming an Actel device

System Verification

You may perform system verification on a programmed antifuse device using the Actel Silicon Explorer diagnostic tool. Refer to the *Silicon Explorer QuickStart* for information about using the Silicon Explorer.

2 Getting Started with Libero IDE

This chapter familiarizes you with the Libero IDE graphical user interface and associated menu commands.

Starting Libero IDE

To start Libero IDE, do one of the following:

- Double-click the Libero icon on your Desktop

- Choose *Libero IDE* from the Libero IDE group in the Programs menu
- Double-click any Libero *.prj file

The Libero Project Manager opens, as shown in Figure 2-1.

To begin using Libero IDE, you must first create a project and then create and/or import source files into your project. For information about creating a project, refer to "Creating a New Project" on page 47.

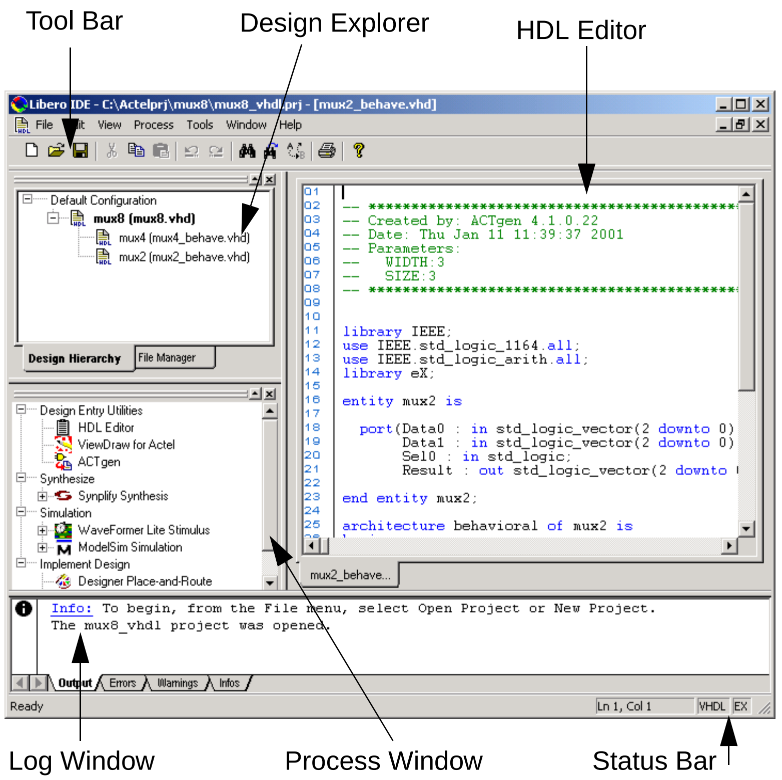
The Libero Project Manager

The Libero Project Manager integrates the needed design tools, streamlines the design flow, manages all design and log files, and passes necessary design data between tools. Libero design tools are listed
in Table 2-1:.

Table 2-1. Libero Tools

Function	Tool	Company
Design Entry (HDL)	HDL Editor	Actel
Design Entry (Schematic)	ViewDraw for Actel	Actel
Synthesis	Synplify or Synplify Lite	Synplicity
Simulation	ModelSim for Actel	Mentor Graphics
Automatic Test Bench Generator	WaveFormer Lite	SynaptiCAD
Automatic Macro Generator	ACTgen	Actel
Place-and-route	Designer Series	Actel
Programming Software	APS/Silicon Sculptor	Actel
In-Silicon Debug	Silicon Explorer II	Actel

The Project Manager consists of the Design Explorer window, the Process window, the HDL Editor window, and the Log window, as shown in Figure 2-1.



```
-- ************************************************
-- Created by: ACTgen 4.1.0.22
-- Date: Thu Jan 11 11:39:37 2001
-- Parameters:
--   WIDTH:3
--   SIZE:3
-- ************************************************


library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
library eX;

entity mux2 is

  port(Data0 : in std_logic_vector(2 downto 0)
       Data1 : in std_logic_vector(2 downto 0)
       Sel0 : in std_logic;
       Result : out std_logic_vector(2 downto

end entity mux2;

architecture behavioral of mux2 is
```

```
Info: To begin, from the File menu, select Open Project or New Project.
The mux8_vhdl project was opened.
```

Figure 2-1. The Libero Project Manager

These windows are explained in detail below.

The Design Explorer

The Design Explorer displays design sources (files) associated with your project. Sources include all design entry files necessary to describe the behavior of your design (such as schematics or HDL source files).

The Design Explorer includes two tab views, the Design Hierarchy tab and the File Manager tab.

Design Hierarchy Tab

Libero IDE continuously analyzes and updates source files and updates the hierarchy. The Design Hierarchy tab displays the structure of the design modules as they relate to each other. The Design Hierarchy tab shows the corresponding file name (the file that defines the block) next to block name in parentheses, as shown in Figure 2-2.

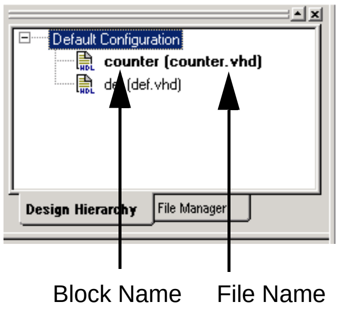


Figure 2-2. Design Hierarchy Tab

Double-click a source in the Design Hierarchy to open it.

If a source is modified and the modification changes the hierarchy of the design, the Design Hierarchy automatically updates to reflect the change.

If you want Libero to re-evaluate the design hierarchy, choose *Refresh* from the Edit menu.

Depending on the block type and design state, several possible options are available from the right-click menu. These options include:

- **Properties**: Displays block properties including, file path, created date, and last modified date.
- **Open Schematic**: Opens the current selected source for editing or viewing
- **Open HDL File**: Opens the HDL file in the HDL Editor
- **Set as Root**: Sets the source as the top level entity
- **Optimize Using Synthesis:** Generates HDL, if necessary, and starts Synplify
- **Create Stimulus**: Generates HDL, if necessary, and starts WaveFormer Lite.
- **Synthesize**: Generates HDL, if necessary, and starts Synplify
- **Run Simulation:** Generates HDL, if necessary, and starts ModelSim for Actel
- **Run Designer:** Starts Designer
- **Run Silicon Explorer**: Starts Silicon Explorer for real time silicon programming
- **Run Silicon Sculptor**: Starts Silicon Sculptor for device programming

File Manager Tab

The File Manager, as shown in Figure 2-3, displays the project design files. These includes block symbol files, schematic files, HDL files, stimulus files, and design implementation files. Files are grouped by type.

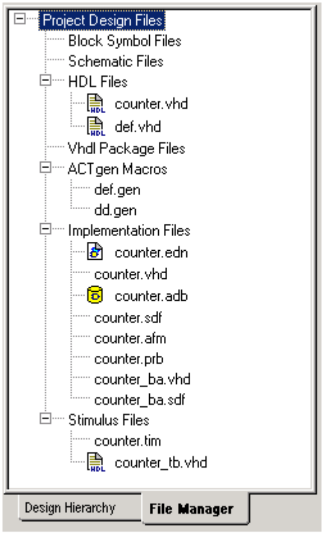


Figure 2-3. the Libero File Manager Tab

Right-clicking a file in the File Manager provides a menu of available options specific to the file type.

Delete files from the project by selecting *Delete from Project* from the right-click menu. Delete files from the project and disk by selecting *Delete from Project and Disk*.

Process Window

The Process window, as shown in Figure 2-4, contains all available tools involved in the design process.

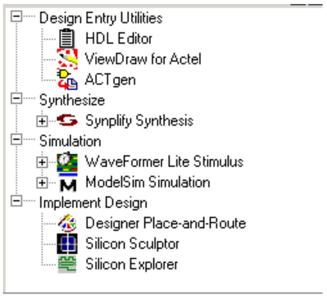


Figure 2-4. Libero Process Window

The current state of your design is also shown in the Process window. Tools are activated at appropriate times in the design process. Tools not yet available are grayed out.

Double-click the tool or application name to start it. The right-click menu lists available processes you can start with the tool. For example, the right-click Model*Sim* Simulation menu includes, depending upon the state of your design, *Run Pre-Synthesis Simulation, Run Post-Synthesis Simulation,* and *Run Post Layout Simulation.*

HDL Editor

Use the Libero HDL Editor for editing your HDL code. Tab windows allow easy navigation to multiple HDL files.

Several commands for the HDL Editor Workspace are available in the File and Edit menus. For more information on the HDL editor, refer to "Design Entry - HDL Sources" on page 57.

Note: To avoid conflicts between changes made in your HDL files, Actel recommends that you use one editor for all of your HDL edits. This eliminates confusion, if more than one editor is open at a given time.

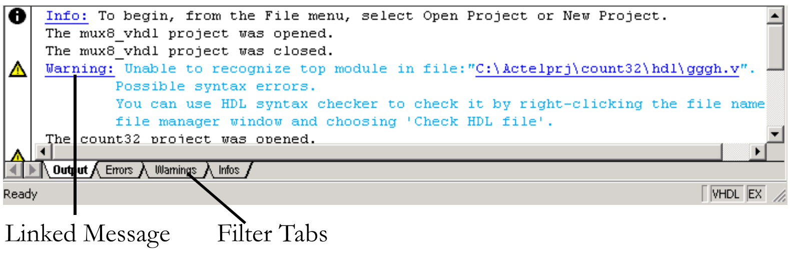
Log Window

For ProASIC and ProASIC$^{\underline{PLUS}}$ families the log window displays notes and warnings. For Antifuse families, the log window displays error, warning, and informational messages. Messages are represented by symbols and color coded, as shown in Table 2-2 :

Table 2-2.

Type	Symbol	Color
Error	🔴	Red
Warning	⚠	Blue
Information	ℹ	Black

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



```
Info: To begin, from the File menu, select Open Project or New Project.
The mux8_vhdl project was opened.
The mux8_vhdl project was closed.
Warning: Unable to recognize top module in file:"C:\Actelprj\count32\hdl\gggh.v".
        Possible syntax errors.
        You can use HDL syntax checker to check it by right-clicking the file name
        file manager window and choosing 'Check HDL file'.
The count32 project was opened.
```

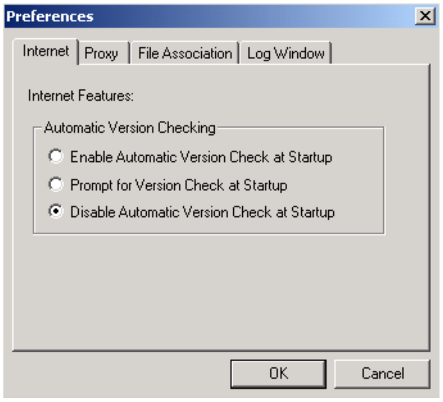
Figure 2-5. Libero Log Window

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

.Setting Preferences

Internet Features

Use the internet tab in the Preferences dialog box to set your automatic version checking preferences, as shown in Figure 2-6. Libero can automatically check if there is a new version at start-up, prompt you before checking, or you can disable version checking entirely.



Internet Features:

Automatic Version Checking

- ○ Enable Automatic Version Check at Startup
- ○ Prompt for Version Check at Startup
- ● Disable Automatic Version Check at Startup

Figure 2-6. Preferences Dialog Box: Internet Tab

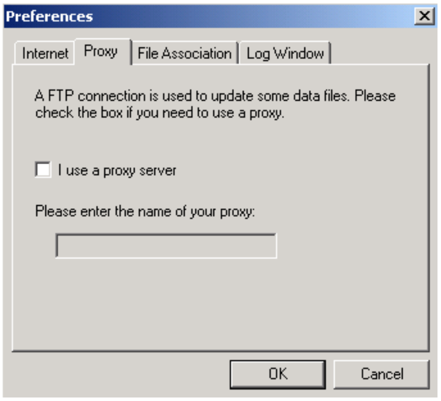
From the File menu, click *Preferences* to set your internet preferences.

Setting Your Proxy

An FTP connection is necessary for the Software Update Check to work. If you use a proxy server due to your firewall configuration, please use the proxy tab to specify the proxy server. Use the Proxy tab in the Preferences dialog box to enter your proxy name, if you use a proxy server.

To set your proxy:

1. **From the File menu, click *Preferences.***
2. **Click the *Proxy* tab, as shown in Figure 2-7.**



A FTP connection is used to update some data files. Please check the box if you need to use a proxy.

☐ I use a proxy server

Please enter the name of your proxy:

Figure 2-7. Setting your Proxy

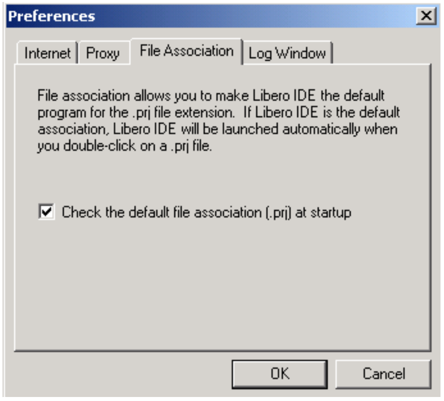
3. **Check the *I use a proxy server box*.**
4. **Type the name of your Proxy and click *OK*.**

File Association

Several programs, including Libero IDE, create files with the .prj extension. Use the File Association dialog box to specify Libero as the default program for files with the .prj extension. Doing so starts Libero whenever a file with the .prj extension is double clicked.

To make Libero the default program:

1. **From the File menu, click *Preferences*. Click the File Association tab, as shown in Figure 2-8.**



File association allows you to make Libero IDE the default program for the .prj file extension. If Libero IDE is the default association, Libero IDE will be launched automatically when you double-click on a .prj file.

[✓] Check the default file association (.prj) at startup

Figure 2-8. File Association Tab

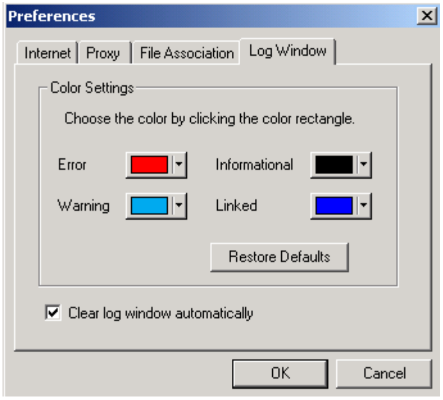
2. **Check the check box to make Libero the default program for files with the .prj extension.**

Log Window

The Log Window tab in the Preferences dialog box allows you to set the default colors used in the log window.

To change the default colors:

1. **From the *File* menu, click *Preferences*.**
2. **Click the Log Window tab, as shown in Figure 2-9**



[✓] Clear log window automatically

Figure 2-9. Log Window Tab

3. **Set your preferences.**
   - Color: To change a default color, click the color box next to the item you want to change and select a new color.
   - Clear log window automatically: Clears the Error, Warning, and Info tabs in the log window each time you perform a significant action. The Output tab is not cleared. By default, this box is checked.
   - Restore Defaults: Restors the default colors.
4. **Click OK.**

Commands

Table 2-3 lists Libero commands and their functions. Frequently used commands are available in the toolbar (page 48)

.

Table 2-3. Libero Commands and Functions

Command	Function
FILE MENU	
New	Creates a new design file
Open	Opens or finds an existing design file
Close	Closes an open design file
New Project	Creates a new Libero project
Open Project	Opens an existing Libero project
Close Project	Closes an open L:ibero project
Import Files	Imports files into Libero
Save	Saves a file or project
Save As	Saves the active file or project with a different name or location
Save All	Save all open files
Print	Prints text in the HDL Editor
Print Preview	Displays a preview
Preferences	Set internet (automatic version checking) and proxy settings
Recent Projects	Lists the last 5 projects saved
Exit	Exit Libero

Table 2-3. Libero Commands and Functions (Continued)

Command	Function
EDIT MENU	
Undo	Reverses your last command
Redo	Repeats your last command
Cut	Removes the selection from the HDL Editor and places it on the Clipboard
Copy	Copies the selection to the clipboard
Paste	Inserts the contents of the Clipboard at the insertion point and replaces any selection
Find	Searches for text in the HDL Editor
Find Next	Searches for next occurrence of text in the HDL Editor
Replace	Replaces text in the HDL Editor
Select All	Selects all text in the active window in the HDL Editor
Refresh	Redraws the window
VIEW MENU	
Toolbar	Toggles toolbar on or off
Status Bar	Toggles status bar on or off
Design Explorer	Opens or closes the Design Explorer window
Process Window	Opens or closes the Process window
PROCESS MENU	
Design Entry Utilities	Starts the HDL Editor, ViewDraw, or ACTgen. Generates HDL or EDIF
Synthesize	Starts Synplify

Table 2-3. Libero Commands and Functions (Continued)

Command	Function
Simulation	Starts ModelSim for Actel or WaveFormer Lite
Implement Design	Starts Designer
TOOL MENU	
Software Update	Notifies you if an update is available from Actel's website
Options	Starts the project setting window, where you can select the device family and set simulation options
WINDOW MENU	
Cascade	Cascades windows in the HDL Editor
Tile	Tiles windows in the HDL Editor
HDL Editor	Toggle between open HDL files
HELP MENU	
Help Topics	Starts online help.
Reference Man-ual	Opens the Libero User's Guide
Search Actel Customer Support	Searches Actel customer support
Actel Website	Opens www.actel.com
About Libero	Displays release information

Toolbar

You can invoke frequently-used commands directly from the Libero toolbar (Figure 2-10).

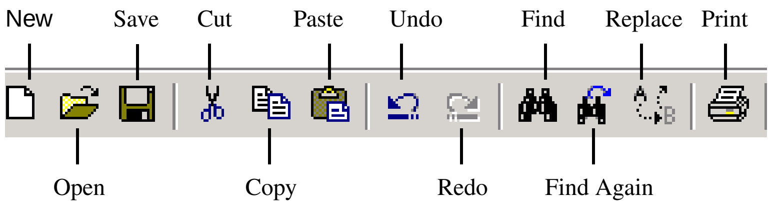


Figure 2-10. Libero Toolbar

Use the View menu to display or hide the toolbar. If you position the mouse pointer over a toolbar button, a short description (called a tool tip) appears next to the button and a longer description appears in the status bar.

Status Bar

The status bar displays information about menu commands and toolbar buttons. In addition, Libero always displays the line and column number of the active HDL file, the HDL language, and the family in the right corner.

3

Managing Your Project

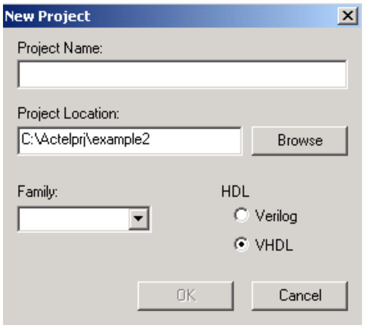
This chapter explains how to create a project, open project and files, import files, and set your project preferences.

Creating a New Project

Begin your project by specifying a project name and directory.

To create a new project:

1. **Start Libero IDE.**
2. **From the File menu, click *New Project.*** This displays the New Project dialog box, as shown in Figure 3-1.



Project Name:
Project Location: C:\Actelprj\example2
Family:
HDL: Verilog [✓ VHDL]

Figure 3-1. New Dialog Box

3. **Enter a Project Name.**
4. **Modify the Project Location field, if necessary**.
5. **Select a family from the Family drop-down list box.**
6. **Select an HDL Language.** Select either Verilog or VHDL, based upon your license.

   Libero creates a new project file <project name>.prj and directory.

Opening an Existing Project

Open an existing project in Libero to continue working on your design.

The 5 most recently opened projects are available from the File menu, under *Recent Projects*.

To open an existing project:

1. **Start Libero IDE.**
2. **From the File menu, click *Open Project*** to display the Open dialog box, as shown in Figure 3-2.

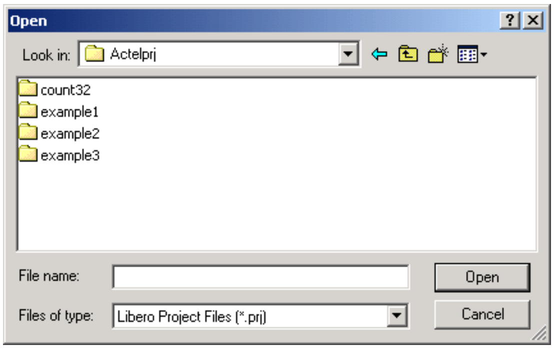
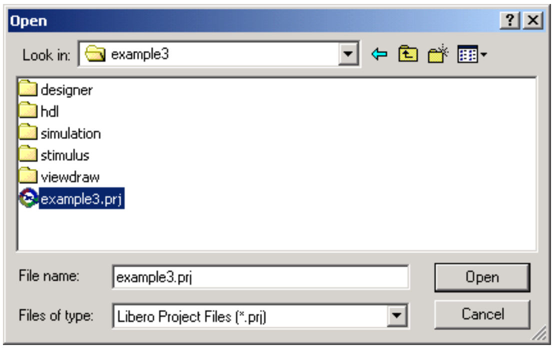


Figure 3-2. Open Dialog Box

3. **Open the desired project directory and select the project file <project name>.prj, as shown in Figure 3-2.**



File name: example3.prj

Files of type: Libero Project Files (*.prj)

Figure 3-3. Opening a Project in Libero

4. **Click *Open*.** The project opens in Libero. Any errors appear in the log window.

Creating Sources for Your Project

After you have opened a project, begin creating and importing sources into the project. Sources are design entry files that contain pertinent information about your design. As you use the Libero integrated tools, the files generated are added as source files to the open project. You can also import sources into your project. Source files appear in the File Manager.

To create a new source for your project:

1. **From the File menu, click *New*.** This displays the New dialog box, as shown in Figure 3-1.

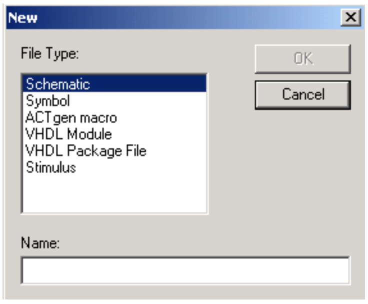


Figure 3-4. New Dialog Box

2. **Select one of the file types and enter a name.** Do not add an extension to the name; Libero adds the appropriate extension. Do not use commas or periods in the file name.
3. **Click *OK*.** Libero starts the corresponding application. When you save your source file, it is automatically saved to the project directory and appears in the File Manager.

Importing Sources into Your Project

You can import avariety of source files into your project, as shown in Table 3-1.

Table 3-1. Source File Types

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

Note: You cannot import a Verilog file into a VHDL project, and vice versa.

Keep and import your VHDL package and behavioral and structural VHDL source files separately. Do not place your VHDL package into your source file. VHDL packages are not supported in the structural flow.

To import a file into your project:

1. **From the File menu, click *Import Files*** to display the Import dialog box, as shown in Figure 3-5.

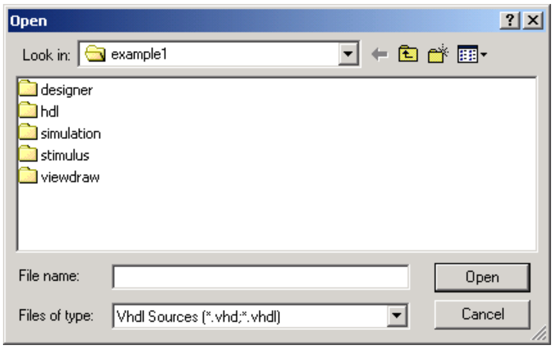


Figure 3-5. Import Dialog Box

2. **Select the file to import and click *Open*.** Filter for files by selecting an option in the Files of type list.

   The file is added to your project and appears in the File Manager.

Deleting Files

Delete files from the project or from the disk from the File Manager tab. Right-click the file and select *Delete from Project* or *Delete from Disk and Project*.

Files deleted from the project are no longer associated with the Libero project, but they remain on your disk. Files deleted from the disk and project are not recoverable.

Project Options

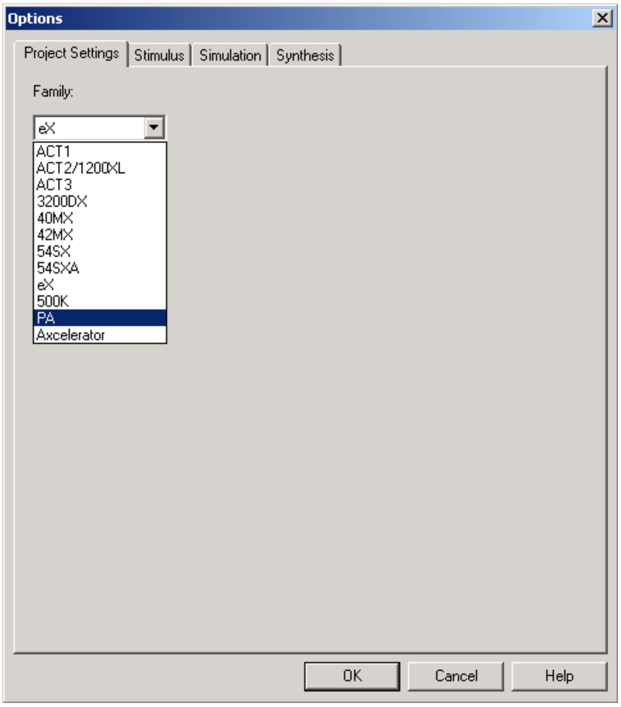
Use the Options Dialog box to specify your project and simulation settings for the currently open project. From the *Tools* menu, click *Options* to open the Options dialog box.

Project Settings

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

To change the device family:

1. **From the Tools menu, click *Options*. The Options dialog box appears, as shown in Figure 3-6.**



Family:

eX

ACT1
ACT2/1200XL
ACT3
3200DX
40MX
42MX
54SX
54SXA
eX
500K
PA
Axcelerator

Figure 3-6. Changing the Device Family

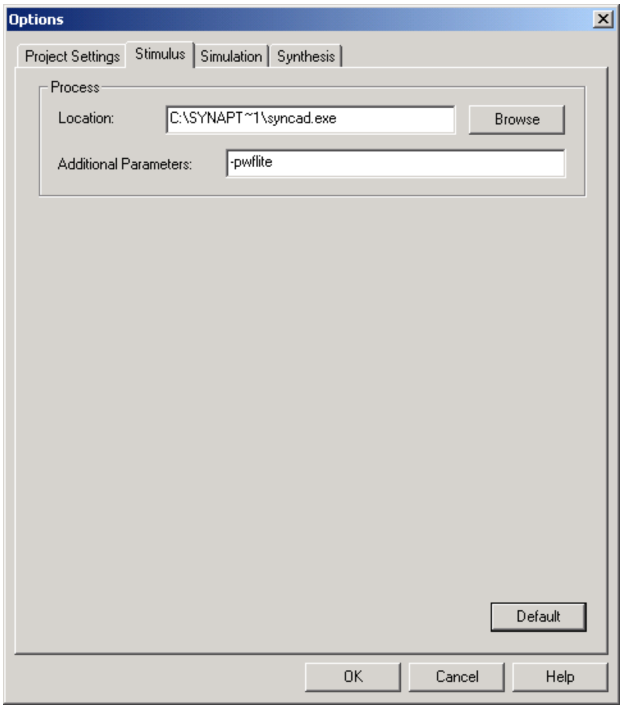
2. **Select a family from the list and click *OK*.**

Stimulus

Use the Stimulus tab in the Options dialog box to view and edit your Stimulus preferences.

To set your stimulus options:

1. **From the *Tools* menu, click *Options* and then the *Stimulus* tab.**



Location: C:\SYNAPT~1\syncad.exe

Additional Parameters: -pwflite

Figure 3-7. Stimulus Options

2. **Set your Stimulus Options and click *OK*.**

   - **Location**: Use to specify the location of WaveFormer Lite. If a full path is not specified, then the tool will be found using the PATH enviornment variable.

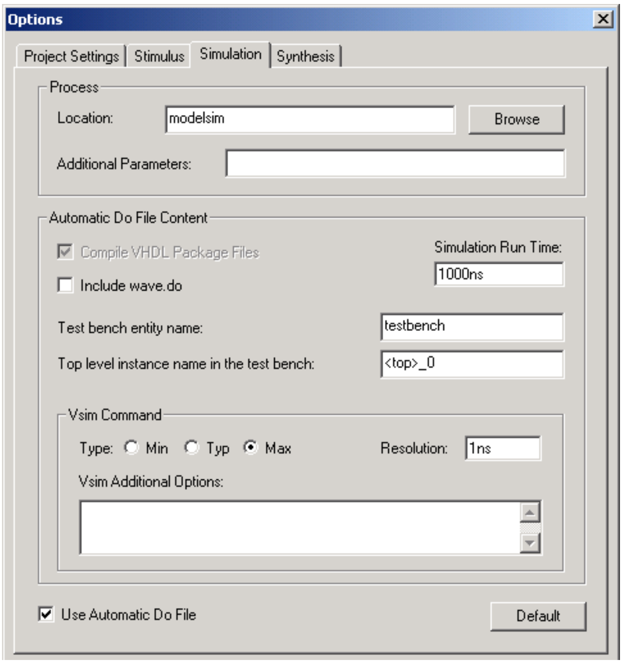
- **Additional Parameters**: Use to specify other settings to pass to WaveFormer Lite. Typically, it is not necessary to modify this field.
- **Default**: Click Default to return to the factory settings.

Simulation Settings

The Simulation tab in the Options dialog box is used to specify the location of ModelSim for Actel and set your simulation preferences.

To set your simulation options:

1. **From the *Tools* menu, click *Options* and then the *Simulation* tab.**



Location: modelsim

Additional Parameters:

[✓] Compile VHDL Package Files

Simulation Run Time: 1000ns

[ ] Include wave.do

Test bench entity name: testbench

Top level instance name in the test bench: <top>_0

Type: Min Typ [✓ Max]

Resolution: 1ns

Vsim Additional Options:

[✓] Use Automatic Do File

Figure 3-8. Setting your Simulation Options

2. **Set your options.**

   - **Location**: Use to specify the location of ModelSim. If a full path is not specified, then the tool will be found using the PATH enviornment variable.
   - **Additional Parrameters**: Use to specify other settings to pass to ModelSim. Typically, it is not necessary to modify this field.
   - **Default**: Click Default to return to the factory settings.
   - **Include wave.do:** Check this to execute the wave.do file. Use the wave.do file to customize the ModelSim Waveform window display settings.
   - **Use automatic do file**: This setting creates a run.do file that is used to specify the files to be compiled and to start the simulation. By default it is checked. This setting is recommended for most users. Uncheck this box only if you do not want Libero IDE to initialize ModelSim for Actel.
   - **Compile VHDL Package files**: If checked, VHDL package files are compiled in ModelSim for Actel. Default is on. Uncheck this if you do not want to compile the VHDL package files.
   - **Simulation Run Time**: Specify how long the simulation should run in ns. If the value is 0, or if the field is empty, there won't be a run command included in the run.do file.
   - **Test bench entity name**: Specify the name of your test bench entity name. Default is "testbench," the value used by WaveFormer Lite.
   - **Top Level instance name in the test bench**: Default is <top_0>", the value used by WaveFormer Lite. Libero replaces <top> by the actual top level macro when ModelSim for Actel is run.
   - **Vsim Command Type**: Select Min, Typical (Typ), or Max.
   - **Resolution**: Default is 1ns.
   - **Vsim additional options**: Text entered in this field is added to the vsim command.

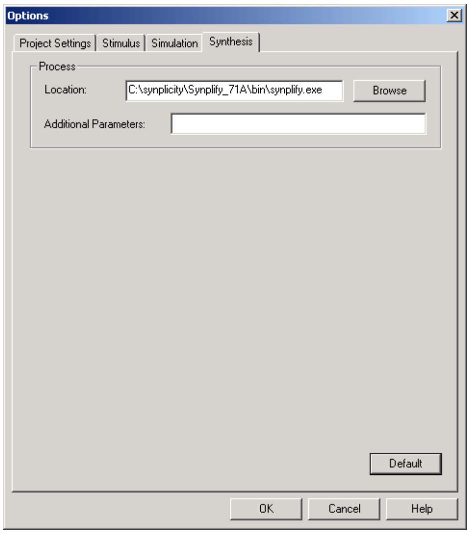
3. **After selecting your Simulation Options, click *OK*.**

Synthesis

Use the Synthesis tab in the Options dialog box to set your synthesis options.

To set your synthesis options:

1. **From the *Tools* menu, click *Options* and then the *Synthesis* tab.** The Synthesis options dialog box appears, as shown in Figure 3-9



Location: C:\synplicity\Synplify_71A\bin\synplify.exe

Additional Parameters:

Figure 3-9. Synthesis Options Dialog Box

2. **Set your synthesis options and click OK.**
   - **Location**: Use to specify the location of Synplify. If a full path is not specified, then the tool will be found using the PATH enviornment variable.
   - **Additional Parrameters**: Use to specify other settings to pass to Synplify. Typically, it is not necessary to modify this field.
   - **Default**: Click Default to return to the factory settings.

4

Design Entry - HDL Sources

Libero supports VHDL and Verilog for the creation of HDL source files.

Creating New HDL Sources

Use Libero’s HDL Editor to create HDL source files for your project.

Starting the HDL Editor:

1. **Open your project.**
2. **From the File menu, click New.** This displays the New dialog box, as shown in Figure 4-1.

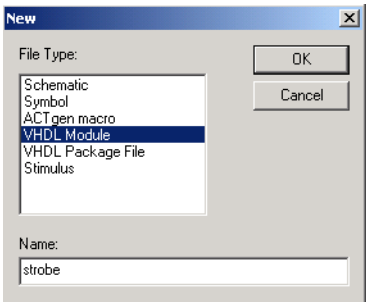
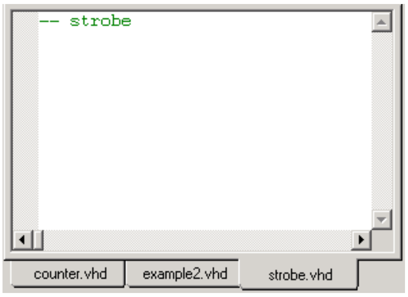


Figure 4-1. New Dialog Box

3. **Select an HDL module (VHDL or Verilog) and enter a file name in the Name field.**
4. **Click *OK*.** Do not enter a file extension; Libero adds one for you.

Your new file opens in the HDL Editor. You can open multiple HDL files at a time, as shown in Figure 4-2.



```
-- strobe
```

Figure 4-2. HDL Editor

5. **After creating your HDL file, save your file to the project by clicking *Save* from the File menu.** Your HDL file is saved to your project, appearing in the File Manager, as shown in Figure 4-3.

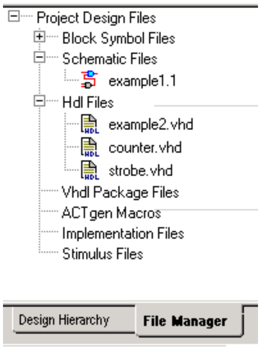
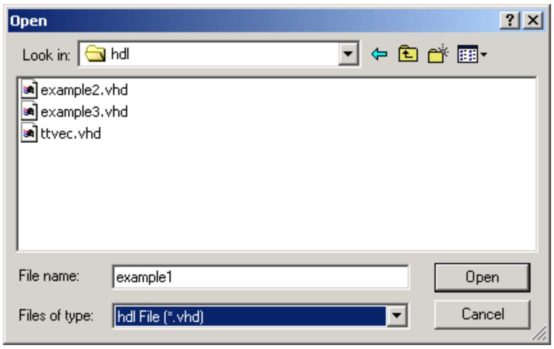


Figure 4-3. Saved HDL Files in the File Manager

Opening Existing HDL Sources

To open an existing HDL source:

1. **Double-click the HDL file in the File Manager, or from the File menu, click *Open*.** This displays the Open dialog box, as shown in Figure 4-4.



File name: example1

Files of type: hdl File (*.vhd)

Figure 4-4. Open Dialog Box

2. **Select your file and click *Open*.** Filter for HDL files by selecting HDL File in the File of Type drop-down list box. Libero opens your file in the HDL Editor.

Using the HDL Editor

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

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

Editing

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

Saving

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

Printing

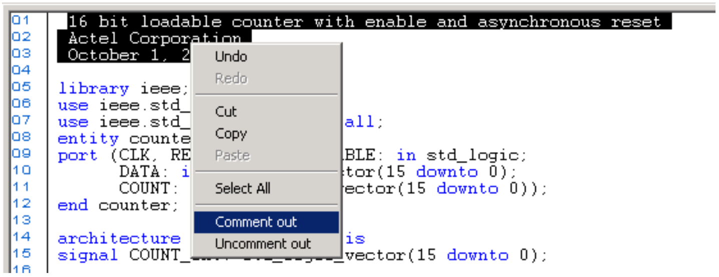
Print and Print Preview functions are available from the File menu and the toolbar.

Commenting Text

Comment text as you type, or comment out text by selecting a group of text and applying the *Comment* command.

To comment out text:

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



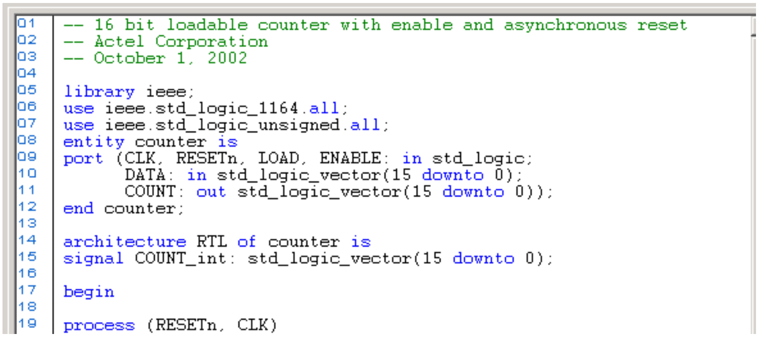
```
 16 bit loadable counter with enable and asynchronous reset
 Actel Corporation
 October 1, 2

library ieee;
use ieee.std_
use ieee.std_                 all;
entity counte
port (CLK, RE                 BLE: in std_logic;
      DATA: i                 tor(15 downto 0);
      COUNT:                  ector(15 downto 0));
end counter;

architecture                  is
signal COUNT_                 vector(15 downto 0);
```

Figure 4-5. Commenting Text in the HDL Editor

The selected text is commented out.



```
-- 16 bit loadable counter with enable and asynchronous reset
-- Actel Corporation
-- October 1, 2002

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is
port (CLK, RESETn, LOAD, ENABLE: in std_logic;
      DATA: in std_logic_vector(15 downto 0);
      COUNT: out std_logic_vector(15 downto 0));
end counter;

architecture RTL of counter is
signal COUNT_int: std_logic_vector(15 downto 0);

begin

process (RESETn, CLK)
```

Figure 4-6. Commented Text

To uncomment text, select the text. From the right-click or Edit menu, click *Uncomment.*

HDL Syntax Checker

Use the HDL Syntax Checker validate an HDL file after editing the HDL code.

To run the syntax checker:

1. **In the Libero File Manager, right-click an HDL file and select *Check HDL*,**as shown in Figure 4-7.

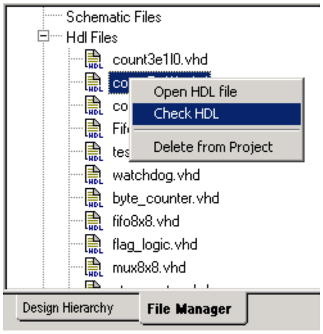


Figure 4-7. Running the Syntax Checker

The syntax checker parses the selected HDL file and looks for typographical mistakes and syntactical errors. Warning and error messages for the HDL file appear in the Libero Log Window.

Adding ACTgen Macros

To add an ACTgen macro to your Libero project:

1. From Libero, start ACTgen by selecting *New* from the File menu.
2. Generate macro to an HDL netlist from ACTgen.
3. Instantiate this macro into your HDL code.

Creating a Schematic Symbol from an HDL Source

To create a schematic symbol from an HDL source, right-click the block in the Design Hierarchy window and select *Create Symbol*, as shown in Figure 4-8.

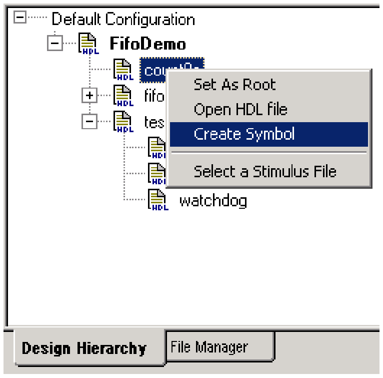


Figure 4-8. Creating Symbols

The symbol file appears in the File Manager, under Block Symbol Files, as shown in Figure 4-9.

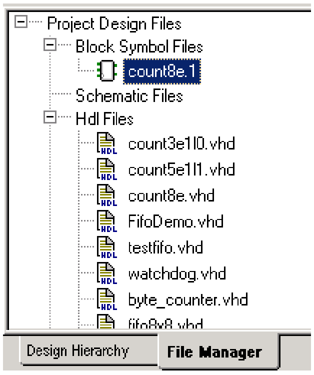


Figure 4-9. Block Symbol Files in the File Manager

5

Design Entry - Schematic Sources

Should your project include schematic sources to define your design, use ViewDraw for Actel to create new schematic sources.

ViewDraw for Actel

ViewDraw for Actel is a customized schematic editor from Mentor Graphics. ViewDraw for Actel is the Libero integrated schematic entry vehicle, supporting mixed mode entry in which HDL blocks and schematic symbols can be mixed.

Using ePD or DxViewDraw with Libero

Actel does not recommend that you use stand-alone versions of ePD or DxViewDraw from Mentor Graphics with Libero IDE. Do not install these and ViewDraw for Actel on the same PC. ViewDraw for Actel and ePD or DxViewDraw use conflicting environment variables and registry entries. If you must install ePD or DxViewDraw and Libero on the same machine, do not install ViewDraw for Actel during the Libero installation.

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

Creating a Schematic Source

The basic steps for using ViewDraw for Actel to create schematic sources for your project are described here. For details about creating your schematic, please refer to "Using ViewDraw for Actel" on page 70, the *ViewDraw User's Guide,* or ViewDraw's online help.

To create a schematic source:

1. **Open your project in Libero.**
2. **From the File menu, click *New*.** This displays the New dialog box, as shown in Figure 5-1.

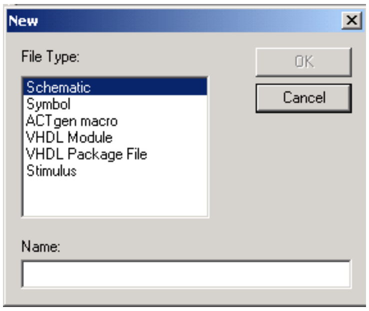


Figure 5-1. New Dialog Box

3. **Select Schematic and type a name for your schematic file in the Name field. Click *OK*.** ViewDraw for Actel starts. Refer to "Using ViewDraw for Actel" on page 70, *ViewDraw User's Guide,* or ViewDraw's online help for information on using ViewDraw for Actel.
4. **Using ViewDraw for Actel, create your schematic.**
5. **Save+Check the schematic.** The *Save+Check* command creates your WIR file. From the File menu, click *Save+Check* to create the required files for netlist generation. When Save and Check is complete, the message "Check complete, 0 errors and 0 warnings in project <name>" appears in the status bar.

   Note: You must select *Save+Check*. Only selecting *Save* will not generate the needed WIR file for that block.
6. **(Optional) Run connectivity checker.** Right-click the schematic file in the File Manager tab and click *Check Schematic*. The connectivity checker checks the connectivity of the wir file. Errors and warnings appear in the log window.

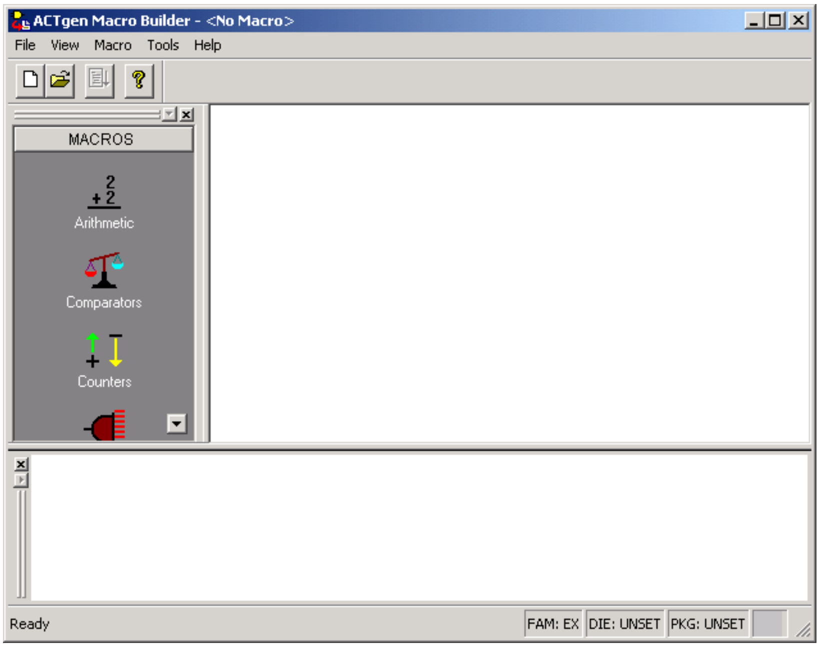
7. **Exit ViewDraw for Actel.** From the File menu, click *Exit*. The schematic is saved to your project in Libero, appearing in both the File Manager and the Design Hierarchy tabs.

Adding ACTgen Macros

The ACTgen Macro Builder allows you to instantly create macros customized to your needs. These macros can them be added to your schematic using ViewDraw for Actel.

To add an ACTgen macro to your Libero project:

1. **From Libero IDE, start ACTgen by selecting *New* from the File menu.**
2. **From the new dialog box, select *ACTgen macro*, type a name, and select *OK*.** ACTgen starts.



3. **Select your macro type from the left Macro list box.** The appropriate options appear. Select a tab and fill in the fields as appropriate.

4. **Click the *Generate* button in the ACTgen toolbar to create your macro.**

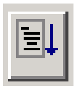
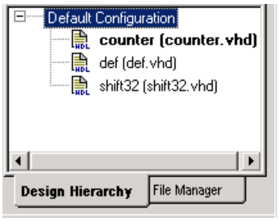
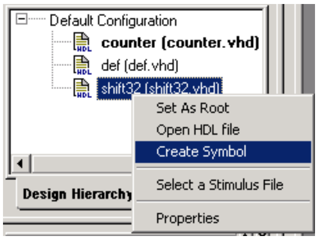


Figure 5-2. Generate Button

The Save As dialog box appears. Leave the default selections and click *Save*. The file is added to your Libero project, appearing in the Design Hierarchy.



5. In the Design Hierarchy window , right-click the ACTgen macro and choose *Create Symbol.*



A message appears in thelog window letting you know the symbol has been created. The symbol also appears in the the File Manager, under Block Symbol files.

6. **Open ViewDraw.** The new symbol is listed in components. You can now instantiate this component in your schematic.

ViewDraw for Actel Schematic Guidelines

When creating your schematics, please follow the guidelines in this section

Using Hierarchical Connectors

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

I/O Pads

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

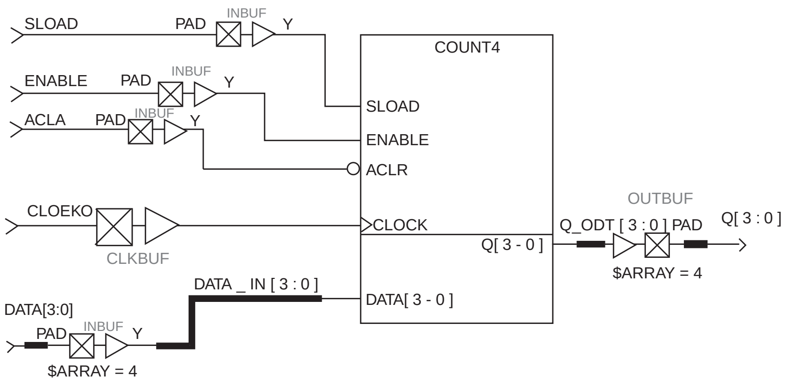


Figure 5-3. Hierarchical Connectors with I/O Pads

Naming Conventions

1. Project names should be less than 8 characters.

   "my_top" is acceptable

   "my_top_level" is not

2. Do not use spaces in:
   - Project names
   - Instance names
   - Net names
   - Port names
3. Do not use any special characters in any of your naming, such as:

   ~ ! @ # $ % ^ & * ( ) = + { } | \ / < > ? ` ' " ", . or spaces.
4. The "inverted" net property is not supported.
5. If you want to rip out scalar bits from a bus, use [] for scalar bit naming. For example, Bus[15], Bus[14], …, Bus[0].
6. Do not use numbers at the beginning or the end of any names. ViewDraw for Actel regards Bus[1] as equivalent to Bus1.

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

   If you want to use numbers to distinguish related nets, numbers can be used followed by letters at the end, for example: NET1N, or Bus1A[15:0].
7. Multi-dimensional busses are not supported. For example, do not use naming "Bus[0:3][0:3]" in ViewDraw for Actel.

Updating Symbols

If you need to change the ports on a block you need to update the corresponding symbol. To update a symbol for a specific block, follow the steps below:

1. **In ViewDraw for Actel, delete all the symbol instantiations from the block's parent level. I**n ViewDraw for Actel, from the File menu, click Save + Check.
2. **In Libero IDE, from the File Manager, delete the symbol from the Project and Disk.** Right-click the symbol file and choose Delete from Project and Disk.
3. **Make the desired modifications to the block. For a schematic block , from the ViewDraw File menu, click *Save* + *Check*. For an HDL block, save to disk.**

4. **From Libero IDE, select the Design Hierarchy tab.** Right-click the block and select *Create Symbol* from the right-click menu.
5. **Instantiate the block symbol back into your schematic.**
6. **Check the basic connectivity of your schematic from Libero IDE.** Right-click the schematic file in the File Manager, and select *Check Schematic*.

Opening a Schematic Source File

To open a schematic source:

1. **Open your project in Libero.**
2. **Open the source file.** Double-click the schematic file in the File Manager or Design Hierarchy windows. ViewDraw for Actel opens with the file loaded. Refer to"Using ViewDraw for Actel" on page 70, the *ViewDraw's User's Guide*, or ViewDraw's online help for information on using ViewDraw for Actel.
3. **Create a WIR file.** From the File menu, click *Save+Check* to create the required files for netlist generation. When Save + Check is complete, the Status Bar will say "Check complete, 0 errors and 0 warnings in project <name>."

   Note: You must select *Save +Check*. Only selecting *Save* will not generate the needed WIR file for that block.
4. **Exit ViewDraw for Actel.** From the File menu, click *Exit*. The schematic is saved to the project, appearing in both the File Manager and Design Hierarchy tabs. Your schematic file is updated in Libero.

Synthesizing Schematic Sources

Synthesize your schematic design if you wish to optimize or automatically insert pads. This is an optional step for structural HDL and/or schematic only design projects. For mix flow designs, you must synthesize the design.

Note: You must include hierarchical connectors in your schematic.

Simulating Schematic Sources

If you want to perform functional simulation using ModelSim for Actel, you must first create a test bench using WaveFormer Lite. Refer to "Creating a Test Bench" on page 151 and "Simulation" on page 171.

Using ViewDraw for Actel

ViewDraw for Actel is the Libero schematic entry tool. This section describes how to use ViewDraw for Actel.

Starting ViewDraw

1. **Open your project in Libero.**
2. **From the File menu, click *New*.** This displays the New dialog box, as shown in Figure 5-4.

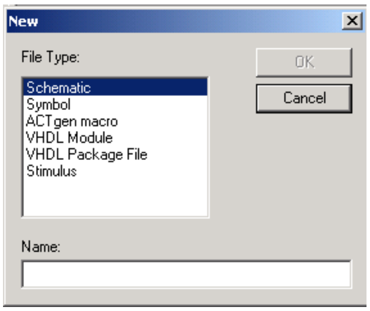
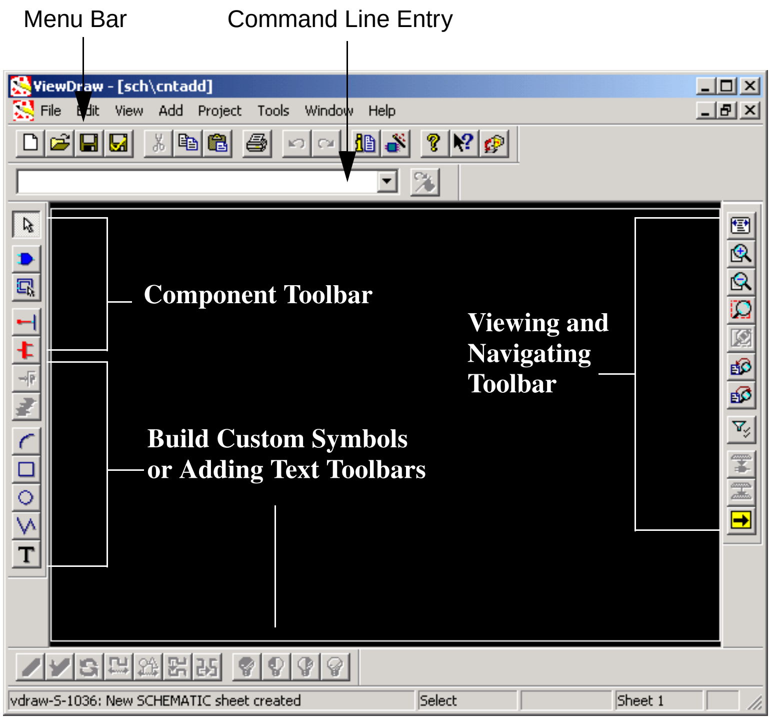


Figure 5-4. New Dialog Box

3. **Click *Schematic* and type a name for your schematic file in the Name field. Click *OK*.** ViewDraw for Actel starts, as shown in Figure 5-5.


Menu Bar
Command Line Entry
ViewDraw - [sch\cntadd]
File Edit View Add Project Tools Window Help
Component Toolbar
Viewing and Navigating Toolbar
Build Custom Symbols or Adding Text Toolbars
Select
Sheet 1

vdraw-S-1036: New SCHEMATIC sheet created

Figure 5-5. ViewDraw for Actel

Now you are ready to create a new schematic by adding components that represent design primitives or by placing components of symbols that you have created or modified on to the schematic sheet. Once you have placed the components add nets and busses to form connections between the components, (see "Adding Nets and Buses for Connectivity" on page 84 for more information.)

Toolbars

ViewDraw for Actel contains 4 toolbars: Standard, Object, Transform, and View. Complete descriptions of each toolbar button are available in ViewDraw online help.

Standard Toolbar

Use the Standard toolbar for your basic project functions.

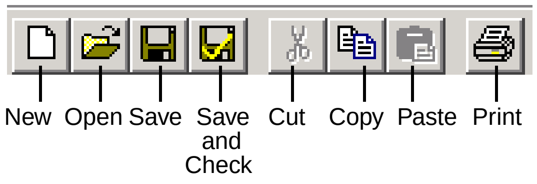


Figure 5-6. ViewDraw Standard Toolbar

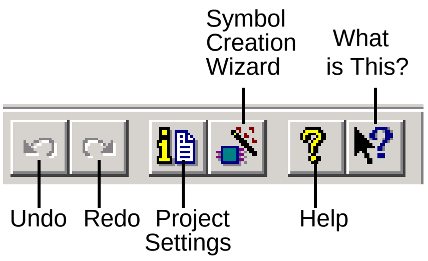


Figure 5-7. ViewDraw Standard Toolbar, continued.

Object Toolbar

Use the object toolbar to add components, create symbols, and add text.

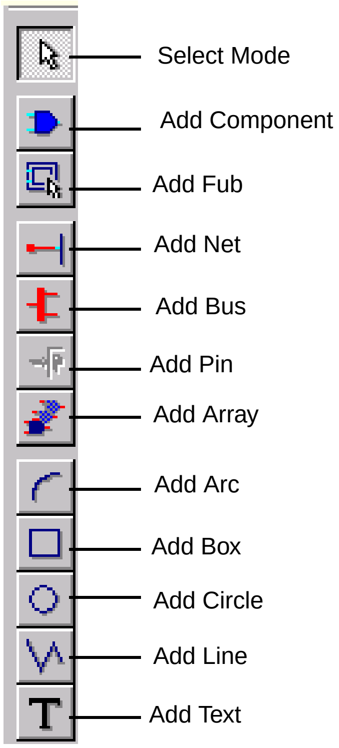


Figure 5-8. Object Toolbar

Transform Toolbar

Use the transform toolbar to create custom symbols.

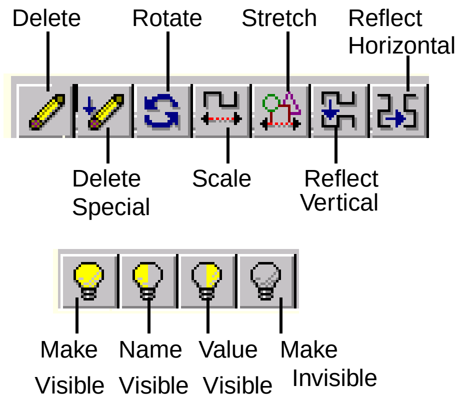


Figure 5-9. Transform Toolbar, continued

View Toolbar

Use the View toolbar to navigate and to change your view.

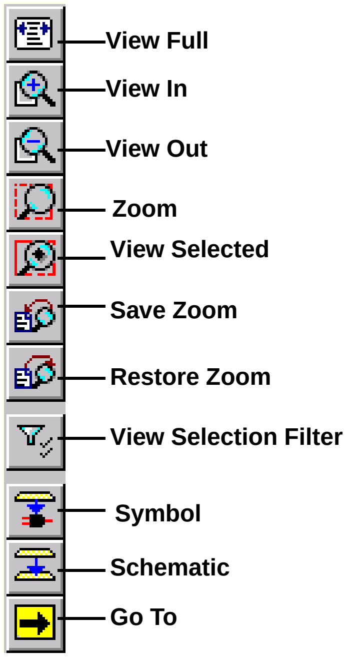


Figure 5-10. View Toolbar

Selecting a Sheet Size

To change the sheet size:

1. **From the Add menu, click *Component*.** The Add Component dialog box appears, as shown in Figure 5-11.

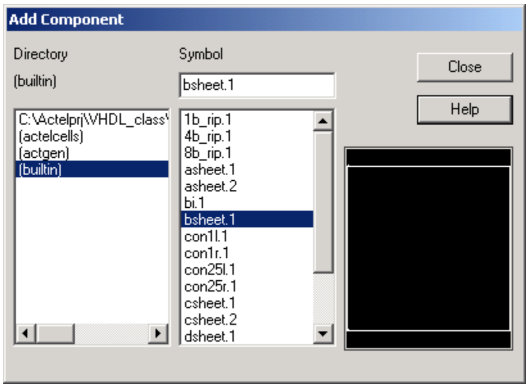


Figure 5-11. Selecting Sheet Size

2. **Select the desired sheet size in Actel ViewDraw Builtin Library.** (For example, asheet.1, bsheet.1)

3. **Drag and drop the sheet into your schematic.**

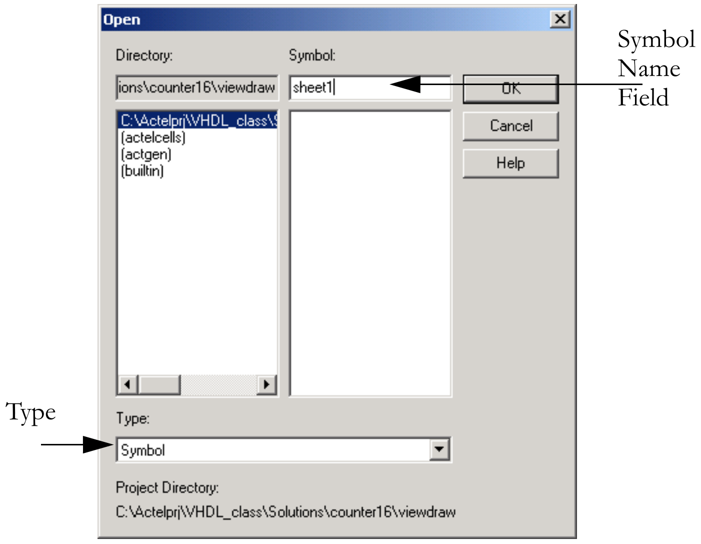
Creating a Sheet Border

You can create a customized sheet border for any size schematic. After you prepare the sheet border, it appears automatically on new schematics.

The Builtin Library contains pre-built sheet borders. If you want to use one of the pre-built sheet borders, select the ASHEET, BSHEET, CSHEET, etc. symbol from the builtin library. Or, you can use the following steps to create and add a custom sheet border.

To create a sheet border:

1. **From the File menu, click *Open*.** The Open dialog box appears.
2. **Choose *Symbol* as the Type, as shown in Figure 5-12.**



Directory: ions\counter16\viewdraw

C:\Actelprj\VHDL_class\S
(actelcells)
(actgen)
(builtin)

Symbol: sheet1

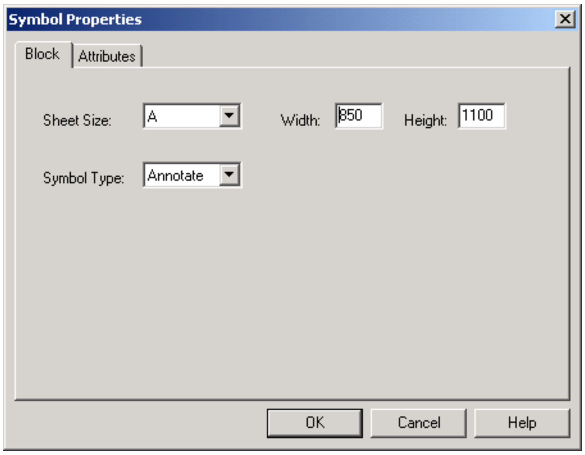
Type: Symbol

Project Directory:
C:\Actelprj\VHDL_class\Solutions\counter16\viewdraw

Figure 5-12. Open Dialog Box

3. **Type in a name for the border in the Symbol field.**
4. **Click *OK* to open the symbol window.**

5. **Double-click the symbol window.** The Symbol Properties dialog box appears, as shown in Figure 5-13.



Sheet Size: A
Width: 850
Height: 1100
Symbol Type: Annotate

Figure 5-13. Changing the Sheet Size Using the Symbol Properties Dialog Box

Change the sheet size by selecting an option in the Sheet List. This size will be your default sheet size. Change the block type to Annotate.

6. **Click *OK* to accept the changes and dismiss the Symbol Properties dialog box.**
7. **Draw the border using the Arc, Box, Circle, and Line commands from the *Add* menu.**
8. **Double-click the symbol window and change the block size to 1 x 1.**
9. **From the File menu, click *Save* to save the border.** The symbol can be saved to any location in your search order.

To add the border to a schematic:

1. **From the File menu, click *Open*.**
2. **Choose *Schematic* as the Type from the Type list.**

3. **Type a name for the schematic in the Schematic field.** For example, asheet.1 or bsheet.1 for a standard size sheet or zsheet.1 for a custom sheet.
4. **From the Add menu, click *Component* or click the *Add Component* toolbar button, as shown in Figure 5-14.**

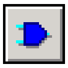


Figure 5-14. Add Component Toolbar button

5. **Click and drag the border component from the list or the symbol preview window to the lower left corner of the schematic. Release the left mouse button to place the symbol.**
6. **From the Project menu, click *Settings* and select the *Block tab.***
7. **Check the *Add Default Sheet* option.**
8. **Change the sheet size option to the same size as your border component (For example, A, B, or Z).**

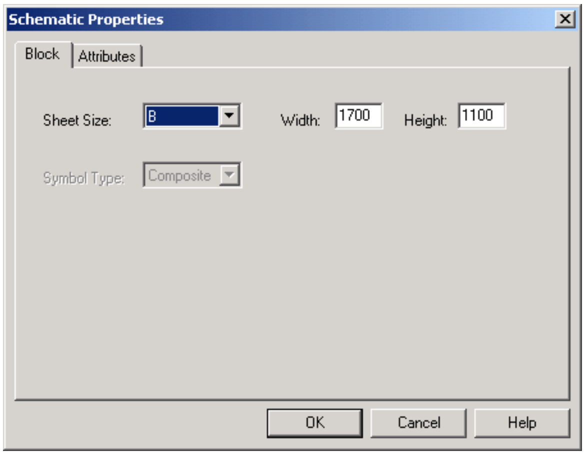
   If you choose Z, you must specify the sheet dimensions in ViewDraw units. These dimensions should match those which you specified as your block size when you drew the border graphic.
9. **From the File menu, click *Save* to save the schematic. ViewDraw saves the schematic to the sch subdirectory.**

   Once the sheet border schematic is created, the DEFSHEET and SHEETSIZE settings in the viewdraw.ini file are modified and all new schematics will have the sheet border. To add the border to existing components, use the Component option from the Add menu to add the component representing the sheet border symbol to the lower left corner of the schematic.

Changing the Sheet Size

To change the schematic sheet size in ViewDraw:

1. **Right-click the schematic background.**
2. **Select *Properties* from the right-click menu.** The Schematic Properties dialog box appears.



Sheet Size: B
Width: 1700
Height: 1100
Symbol Type: Composite

Figure 5-15. Schematic Properties Dialog Box

3. **Choose your preference and click *OK*.**

Showing Time and Date in the Schematic

To show the time and date in the schematic:

1. **Right-click any blank space in the schematic, select *Properties* and click *Attributes* from the right-click menu.**
2. **Add attribute @DATETIME.**
3. **Click *OK*.** The time and date will be update every time you modify and save the schematic.

Adding Components

Components allow you to use a symbol many times in a schematic. Use the *Component* command from the Add menu or the *Add Component* toolbar button to add components to the schematic. The command-line command for this function is **comp**.

To add a component to the active schematic window:

1. **From the Add menu, click *Component* or click the *Component* toolbar button.** The Add Component dialog box appears, as shown in Figure 5-16.

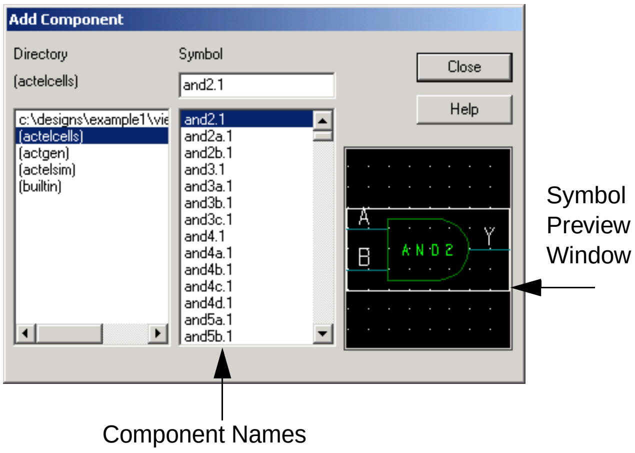


Figure 5-16. Add Component Dialog Box

2. **Select a Directory from the list box.** Select ACTELCELLS for all Actel library cells. ACTELSIM are special purpose libraries and should not be used for selecting components. The BUILTIN library should only be used for special macros such as connectors, bus rippers, and sheet size.
3. **Click and drag the component from either the component list or the Symbol Preview window onto the schematic sheet.**
4. **Click *Cancel* to dismiss the Add Component Dialog box.**

   Note: Pressing *ESC* while dragging the component cancels the operation.

You can keep the Add Component dialog box open as you edit the schematic. This allows you to add components at any time without reopening the Add

Component dialog box for each component you add.

Copying Components

You can copy a component or object to a different location in the active window, in another ViewDraw for Actel window, or another application window.

To copy a component to a different location in the same window:

1. **Click the component or object to select it.**
2. **While holding the CTRL key, select the object again with your mouse and drag the copied component to the new location in the active window and release the mouse button.**

To copy a group of objects to a different location in the same window:

1. **Select the objects you want to copy.** To select multiple objects, hold down the CTRL key while selecting objects, or click and drag over an entire area.
2. **While pressing the CTRL key, drag the selected objects to the new location in the active window and release the mouse button.**

   All other selected objects are copied, maintaining their relative position.

To copy a component or object to another ViewDraw for Actel or application window:

1. **Click a component, or click and drag over a group of components or objects to select an entire group.**
2. **From the Edit menu, click *Copy* or click the toolbar button.** CTRL + C also copies the selected component or object.
3. **Click the window you want to paste this component in.**
4. **From the Edit menu, click *Paste* or click the toolbar button to paste the component or object.** CTRL+V also pastes the copied component or object.

To copy a bitmap picture of component to an application window:

1. **Click a component or click and drag over a group of components or objects to select them.**
2. **From the Edit menu, click *Copy* or click the *Copy* toolbar button.** CTRL+C also copies the selected component or object.
3. **Open and activate the application window that you want to paste the component in.**
4. **From the Edit menu, click *Paste Special* or click the *Paste* toolbar button to paste the component or object.**
5. **Click *OK* on the Paste Special dialog box.**

   Bitmap images show objects as selected. If you want to copy a bitmap without the selection boxes, zoom in so that the section you want to copy fills the screen and perform the copy with nothing selected.

Connecting Components

Once you place components on a schematic, connect them using nets and buses. You construct a net or a bus under a specified routing mode that assists with determining the optimal route path.

The following types of connections are valid:

- Intersecting Connections
- Dangling Connections
- Connections by Abutment
- Connecting Identical Labels

Adding Nets and Buses for Connectivity

Add buses or nets for increased connectivity.

What is a Net?

Use nets to specify an input or output lead from a component pin to any point on an existing net or bus. A net represents an electrical connection. You create nets between component pins, from a single component pin, or between nets.

You can construct a net with one or more segments. If a net has more than one segment, ViewDraw for Actel indicates the segment endpoints by joints at the net vertices. You can connect maximum of four net segments at a net joint.

A net is not the same as a line. A line is only graphical; a net carries a signal.

What is a Bus?

A bus is a collection of nets that can operate as a group or as individual nets within the bus. Create buses between component bus pins, from a single component bus pin, or between nets.

A bus groups related signals. For example, the bus IN[0:7] represents the signals IN[0], IN[1], IN[2], IN[3], IN[4], IN[5], IN[6], and IN[7]. You specify bus names and ranges (widths) using labels.

To add a net to the active schematic:

1. **From the Add menu, click *Net* or click the *Net* Button on the object toolbar.**
2. **Click and drag to form the net, specifying points along the net by clicking the right mouse button or pressing the spacebar.** The current routing mode determines how the connection if formed.

   Note: Nets must begin at a component pin or at an existing net.
3. **Release the left mouse button to specify an ending point for the net.** Once you enter Add Bus Mode, you can add multiple buses without selecting the Add Bus function again.

To add a bus to the active schematic

1. **From the Add menu, click *Bus* or click the *Bus* toolbar button.** (You can also use the b hotkey or the bus command.)
2. **Position the cursor at the originating point of the bus and press the left mouse button.**
3. **Drag the mouse to draw the bus.** The current routing mode (straight, avoidance, or orthogonal) determines how the connection is formed.
4. **Click the right mouse to insert a vertex in the bus.**
5. **Release the left mouse button to specify the ending point for the bus.** Once you enter Add Bus Mode, you can add multiple buses without selecting the Add Bus function again.

Bus Tips and Hints

Soft macros and ACTgen generated macros typically define data inputs as busses, for its symbols. There are several different ways to work with bus structures in ViewDraw for Actel.

Small to Larger

When busses (for instance combined with individual nets) are combined to generate a larger bus, the output and input busses cannot be physically connected on the schematic, as shown in Figure 5-17.

The net names map directly to the symbol names in the order defined (A7 = DATA17, first GND = DATA9, and B7=DATA7.) All inputs on a bus must be driven by some source even if the input is a "don't care."

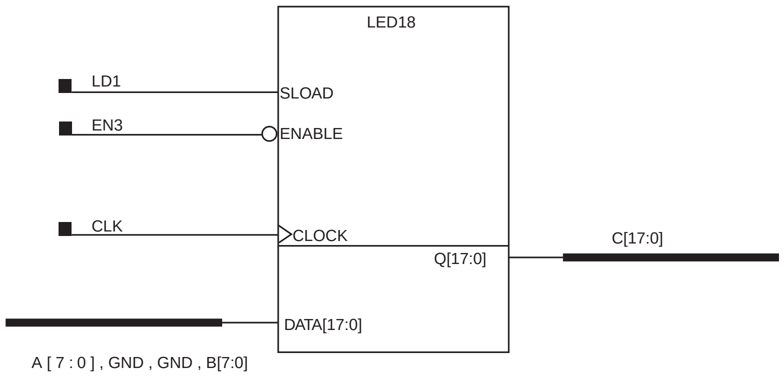


Figure 5-17. Small Bus Feeding Larger Bus

Larger to Smaller

When a large bus feeds into smaller busses, the busses can be physically connected or separated depending on your preference. The inputs can be defined to have any bit sequence of the output and the label should be attached to the bus segment which directly attaches to the bus pin, as shown in Figure 5-18. Unused outputs will not create any errors (C4 and C5 are not used).

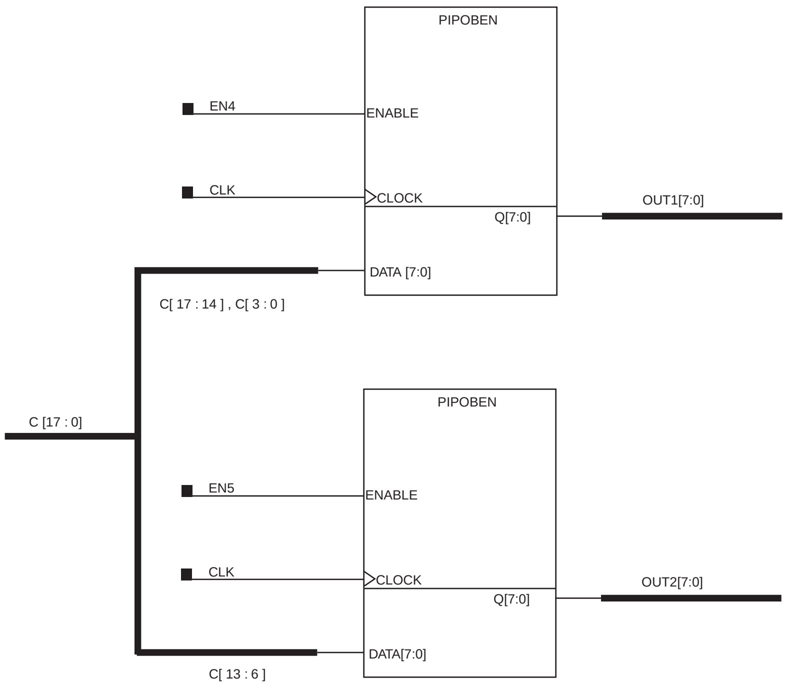


Figure 5-18. Large Bus Feeding Smaller Busses

Terminated to Ground or Terminated to VCC

A bus terminated all to ground or VCC can be defined by labeling the bus with the net names GND or VCC for each bit of the bus, or by using the “Constant Input Reduction” flag as follows.

To use Constant Input Reduction:

1. **Connect the GND or VCC symbol to the input of a buffer using a net.**
2. **Assign the buffer the following attribute:**

   ```
   $Array=bus_width (decimal value)
   ```

   This attribute causes the buffer to be replicated in the netlist by the value of "bus_width."
3. **Label the bus with a name of appropriate width.**

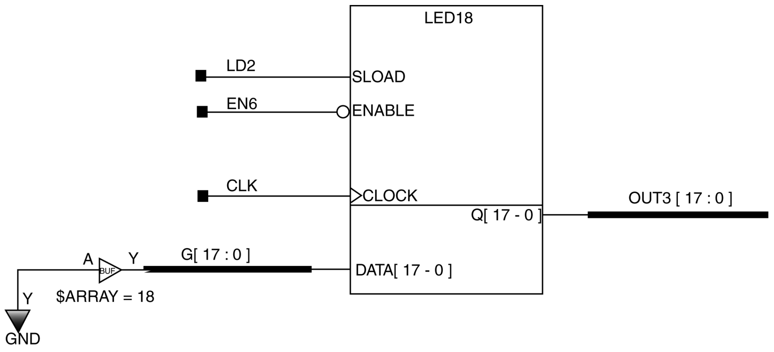


Figure 5-19. Constant Input Reduction

Terminated to Ground and VCC

A bus that has a constant value and is terminated to a combination of VCC and GND values can be defined in a number of ways. You can label the bus with the net names GND or VCC for each bit of the bus; however, for wide busses this can be very cumbersome. An alternative approach is to alias the VCC and GND names to single alphabetical character values,as shown in Figure 5-20.

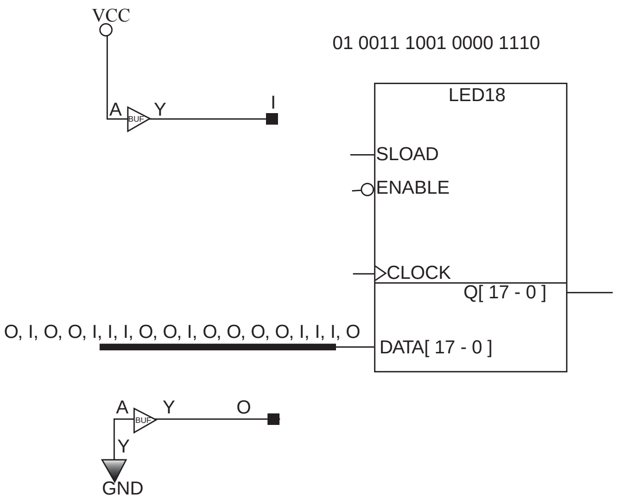


Figure 5-20. Alias Names

To use aliases:

1. **Connect GND and VCC symbols to the input of a buffer using a net.**
2. **Label the buffer output net with an alphabetic character. For example 1 (one) and O (zero).**
3. **Label the input bus using the same names for GND and VCC.**

Another approach is to generate VCC and GND busses and then connect the appropriate value to each bus bit by giving the VCC and GND busses the same names as the input bus.

To use the input bus names:

1. **Connect GND and VCC symbols to the input of a buffer using a net.**
2. **Assign the buffer the following attribute:**

```
$ARRAY=bus_width(decimal_value)
```

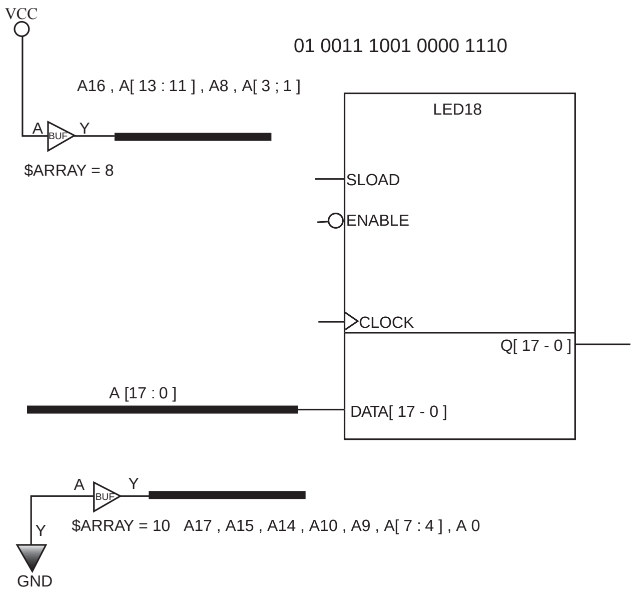


Figure 5-21. Input Bus Names

This attribute causes the buffer to be replicated in the netlist by the value of "bus width."

3. **Label the bus with the appropriate bits of the bus that feeds the input.**

Logic Replication

There are some cases using the ACTgen Macro Builder where functional logic needs to be replicated to drive a bus.

The following example illustrates several options for replicating functional logic. In this example, the tristate enable pin "TRIEN[4:0]" is a 5-bit input bus. The TRIEN pin is controlled by a 3-input AND gate. Heavy loading for this case is not a problem, so the design uses two AND gates to drive the 5 TRIEN inputs. The synchronous load input pin "SLOAD[5:0]" is a 6-bit input bus which is controlled by a two-input pin OR gate. Heavy loading could be a problem for this case, so the design uses 6 OR gates.

The 22-bit register can be readily generated to drive the input data bus simply by applying the

```
$ARRAY=22 attribute.
```

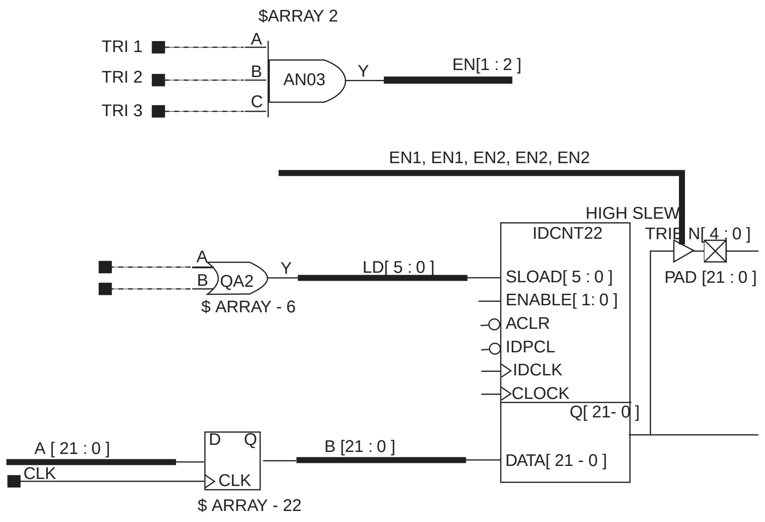


Figure 5-22. Logic Replication

Intersecting Connections

Two nets that cross make a connection only if a round solder dot appears at the crossing. Any incidental crossing of nets or buses from schematic edits does not imply a connection. Net solder dots appear at 3-way or 4-way connections. Bus solder dots appear only at 4-way connections.

To make an interconnection between two nets:

1. **From the Add menu, click *Net* or click the Net toolbar button.**
2. **Click and drag your mouse to form the net, specifying the points along the net by clicking the right mouse button.** The current routing mode determines how the connection if formed.

   Click the right mouse button or the spacebar with left mouse button pressed to insert a vertex in the net.

3. **Repeat steps 1 through 2 to create the intersecting net.**
4. **From the Add menu, click *N*et or click the Net toolbar button.**
5. **Click at the intersecting point of the nets to create the solder joint.**

Dangling Connections

A net that does not connect to a pin or another net ends with a square box to indicate a dangling connection.

You can connect or continue a dangling net by clicking the box while the *Add Net* command is activated.

Note: You can also select a segment and stretch it.

Preserving Dangling Connections

To maintain dangling connectivity when you delete an object, hold down the CTRL key when you select Delete from the Edit menu. Or, right-click the object and select *Delete Special.*

Adding Attributes

Use attributes to compose symbol definitions for interpretation when wirelisting. The wirelist uses the first symbol definition it encounters and interprets the attributes associated with this symbol. Therefore, add attributes at hierarchical levels where the wirelist will use the definition that you intend.

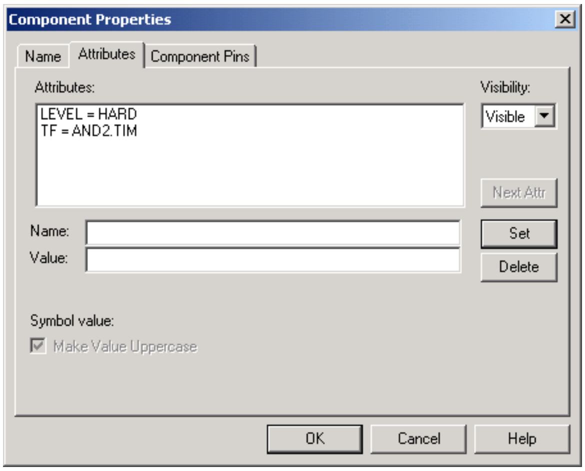
You can add attributes to:

- Symbols (unattached)
- Symbol pins
- Schematics (unattached)
- Components
- Component pins
- Net segments
- Bus segments

You cannot place attributes on boxes, lines, arcs or circles.

To add an attribute:

1. **Double-click the object you want to add the attribute to. The Properties dialog box appears, as shown in Figure 5-23.**
2. **Click the Attributes tab of the dialog box.**



LEVEL = HARD
TF = AND2.TIM

Figure 5-23. Component Properties Dialog Box

3. **Enter the attribute name in the Name field.**
4. **Enter the attribute value in the Value field.**
5. **Select the attribute visibility selection from the Visibility pull-down list.**
6. **Set the REFDES attribute on symbols and the # attribute on symbol pins to invisible because their values are automatically promoted to the component level on the schematic.**
7. **Click *Set* to add the attribute.**
8. **Click *OK* to dismiss the Properties dialog box.**

Bus Rippers

A bus ripper is a symbol that you instantiate in your design to pull signals from a bus. This object — the bus ripper symbol — provides you with an unambiguous reference to selected signals of the bus. You can also use bus rippers to create an alias net name for signals that you specify. If you do not intend to create alias net names, you can extract signals from a bus the traditional way (without using a bus ripper).

Moving Through The Levels of The Hierarchy

You can explore the hierarchy of a schematic using the following options:

To page through the sheets of a schematic:

Use the Page Up and Page Down keys or use the GoTo, Previous Page, or Next Page options from the pop up menu.

To move to a different level or sheet in the schematic or symbol:

1. **Use the Schematic or Symbol options from the pop up menu.**

   Right-click a component in the active window to get the pop up menu. The title bar of the active window indicates the schematic or symbol's name.

   These menus allow you to go to any sheet in your design. To return up in the design, either close the lower layer windows or use the Windows menu to select the desired window.

Customizing the Color Palette

Customizing the color palette allows you to define the 16 colors to use for your screen colors and another 16 colors for project print colors.

To create your own customized color:

1. **From the Project menu, click *Settings*.**
2. **Choose the *Color Palette* tab.**
3. **In the Choose Color field, select the color you want to replace with the custom color.**

4. **Double-click the color in the color box or click the *Edit* button to access the Color dialog box, as shown in Figure 5-24.**

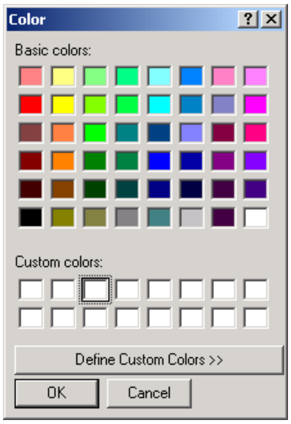
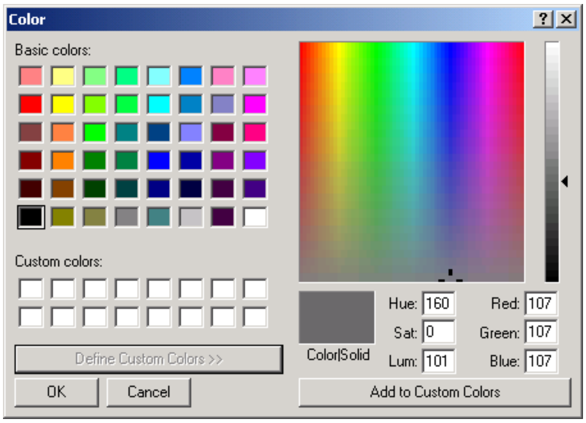


Figure 5-24. ViewDraw for Actel's Color Dialog Box

5. **Select a color to edit by clicking on that color in the Basic Colors or Custom Colors box.**

6. **Click the *Define Custom Colors* button.** The Colors Dialog box expands, as shown in Figure 5-25.



Hue: 160
Sat: 0
Lum: 101
Red: 107
Green: 107
Blue: 107

Figure 5-25. Defining Custom Colors

7. **Change the luminosity of your color by clicking the mouse inside the luminosity strip, dragging the cursor to affect color brightness.** (The LUM box indicates the new luminosity value; the defined color box displays the new color.)

8. **Change the color qualities; Hue, Saturation, Red, Green, and Blue.** (Do this by dragging the crosshairs in the color variation box. As you move the mouse, the values for each of the qualities change depending on the position of the cross-hairs. The color defined by your choices is displayed in the color box.)

9. **Click *Add to Custom Colors* to add the color to your custom color option.**

10. **Click *OK* to dismiss the Color Palette dialog box.**

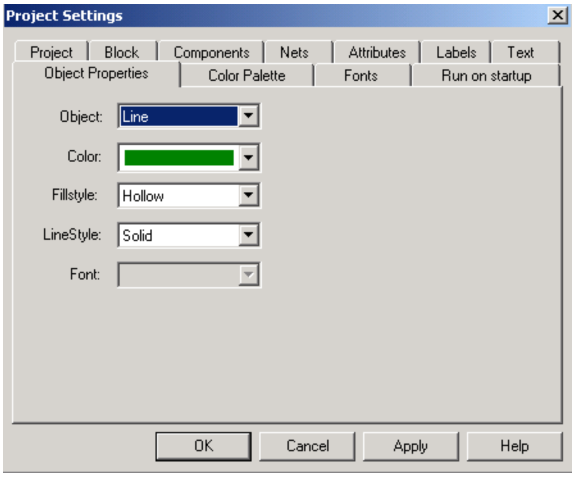
11. **Click *OK* to accept the new color palette and dismiss the Project Settings dialog box.**

Changing Object Colors

You can change how objects are displayed on your screen using the Settings command from the Project menu.

To change the color settings for a graphical object:

1. **From the Project menu, click *Settings*.**
2. **Click the *Object Properties* tab as shown in Figure 5-26.**



Object: Line
Color: 🟩
Fillstyle: Hollow
LineStyle: Solid
Font:

Figure 5-26. Object Properties Tab in Project Settings Dialog Box

3. **In the Object field, choose the object type whose color you want to change.**
4. **In the Color field, choose the color for the object.**
5. **In the Fill Style field, choose the fill style for the object.**
6. **In the Line Style field, choose the line style for the object.**
7. **Click *OK*.**

   Note: If the object is a text object, you can change the only the color and font style for the object.

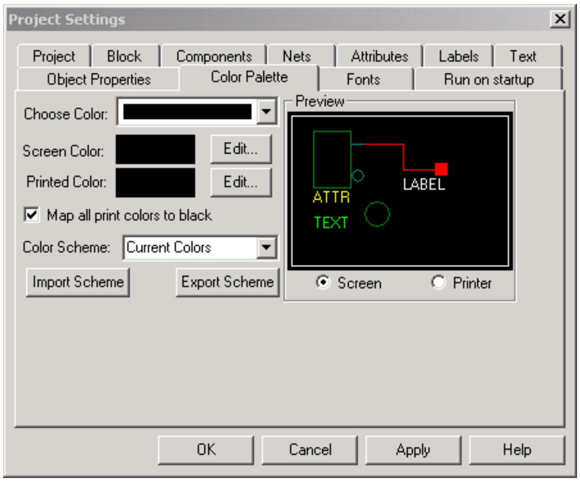
Changing Viewing and Printing Colors

ViewDraw for Actel has a default color scheme for you to use. If you have a preferred color scheme, you can change the colors. You can alter the colors of the default color scheme, or you can change the color scheme altogether.

Note: The background color for your schematic defaults to black. You can change the background color by changing the screen color for white.

To alter the colors in the default color scheme:

1. **From the Project menu, click *Settings* to open the Settings dialog box, as shown in Figure 5-27.**



Project Settings

Tabs: Project, Block, Components, Nets, Attributes, Labels, Text, Object Properties, Color Palette, Fonts, Run on startup

Choose Color:

Screen Color: Edit...

Printed Color: Edit...

☑ Map all print colors to black

Color Scheme: Current Colors

Import Scheme   Export Scheme

Preview: (● Screen ○ Printer)

Figure 5-27. Settings Dialog Box, Color Palette Tab

2. **Choose the *Color Palette* tab.**
3. **Choose a color from the Choose Color list.**

   This list includes all the colors that are available to you. The last color in the list is your windows default background color.

   You can also change the screen color by clicking the *Edit* button next to the Screen Color list. A palate of colors is displayed.

4. **Select the color want and click *OK*.**

   This changes the color you see on your screen for all objects and text mapped to the color in the Choose Color field.

5. **To select a printer color, click the *Edit* button to the right of the Printed Color option. Select the color want and click *OK*.**

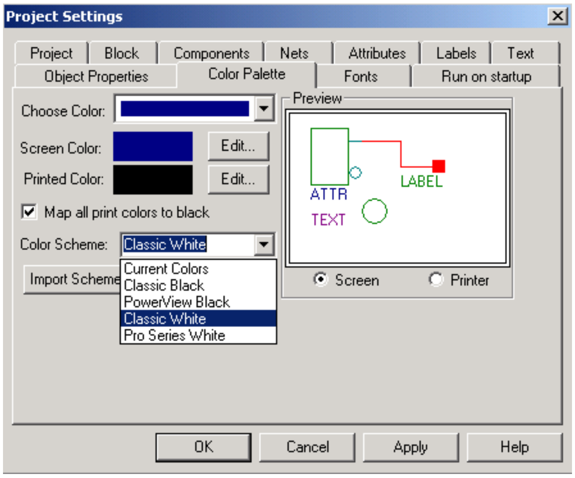
This changes the color that prints for all objects and text mapped to the color in the Choose Color field.

6. **Click *OK* to close the Settings dialog box.**

   Keep in mind that non-color printers map colors to grayscale. In some cases, printers map colors to white so that you cannot see them. You can eliminate problems with color mappings by selecting the Map all print colors to black option.

To change the color scheme altogether:

1. **From the Project menu, click Settings.**
2. **Choose the Color Palette tab.**
3. **Choose a different color scheme from the Color Scheme list, as shown in Figure 5-28.**



Choose Color: 🟦
Screen Color: 🟦
Printed Color: ⬛
[✓] Map all print colors to black
Color Scheme: Classic White
Current Colors
Classic Black
PowerView Black
Classic White
Pro Series White
Preview: [✓ Screen] Printer

Figure 5-28. Selecting a Color Scheme in ViewDraw for Actel

4. **Click *OK*.**

Printing

Before printing from ViewDraw for Actel for the first time, you must connect the printer to the computer or network, install a printer driver, and select the printer to use. If you have not completed these tasks, please do so before attempting to print your schematic or symbol.

Print the active window or a specified schematic sheet using the Window and Sheet options in the Print dialog box. You can also print an entire design by clicking *Print Project* in the File menu.

Because colors that are easy to work with during a ViewDraw for Actel session are not always the best colors to use for printing, ViewDraw for Actel allows you to define a viewing color and a printing color for each object. Refer to "Changing Viewing and Printing Colors" on page 97 for information about defining colors for graphical objects, components, text, and annotation objects.

To print a document:

1. **From the File menu, click *Print*.**
2. **Fill in the appropriate fields in the Print dialog box.** For more information about Print dialog box options, click the Help button on the dialog box.

   Alternative: If you want to use default print settings, click the *Print* toolbar button, as shown in Figure 5-29.

Figure 5-29. Print Toolbar Button

6 Synthesis

Synplicity's Synplify and Synplify Lite are the Libero IDE integrated synthesis tool. Synplify can effectively synthesize VHDL or Verilog language designs to create EDIF netlists. For detailed information about using Synplify tools to get the best results, please refer to the Synplify and Synplify Pro User's Guide or Synplify on-line help. This information can be accessed from the Synplify product, or from the following web sites at Synplicity:

http://www.synplicity.com/downloads/index.html

http://www.synplicity.com/literature/index.html

This chapter contains a brief description of using Synplicity with Libero IDE. For more specific details on using Synplicity, refer to Synplicity's Online Help or User's Guide, available by selecting *Help* from Synplicity's Help menu.

Synplify Lite and Synplify for Actel

Synplify Lite comes with Libero Silver and Gold editions. Libero Platinum comes with Synplify for Actel. The differences are summarized in Table 6-1.

Table 6-1. Synthesis Options

Feature	**Libero Silver (Synplify Lite)**	**Libero Gold (Synplify Lite)**	**Libero Platinum (Synplify for Actel)**
Synplicity's Proprietary Behaviour Extracting Synthesis Technology (BEST™) Algorithms	•	•	•
Integrated Module Generation and Mapping	•	•	•
SCOPE Multi-Level Design Constraints	No	No	•
Comprehensive Language Support	•	•	•
Language-Sensitive Editor	•	•	•
Intuitive Use Model with Intelligent Defaults	•	•	•
Direct Synthesis Technology	•	•	•
Advanced Register Detection	•	•	•
Hierarchy Browser Display	No	No	•
Tcl Scripting	No	No	•
I/O Insertion Control (1)	No	No	•
Netlist Hierarchy	Flat	Flat	•
HDL Analyst Compatibility	No	No	2
Applies to all Actel FPGA Product Sizes	Max 10k Gates, or smallest member of device family	Max 50k Gates, or smallest member of device family	All Devices

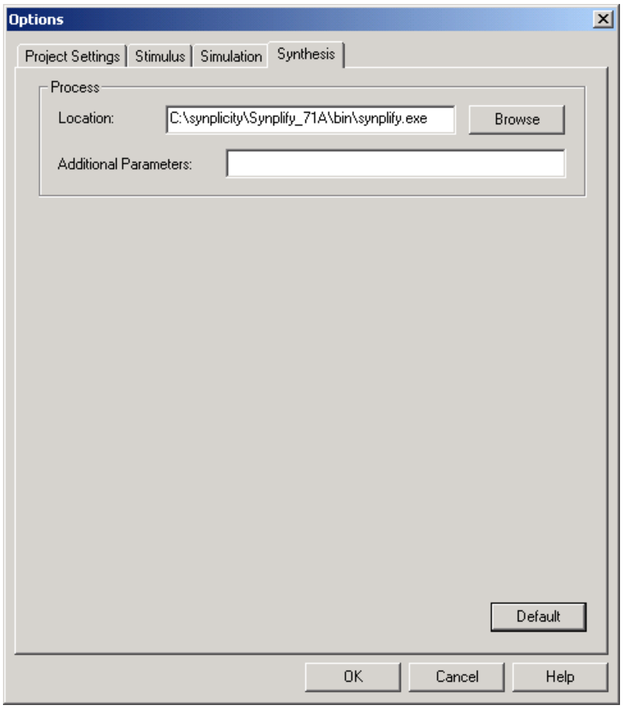
1. Cannot be disabled, user selectable in Platinum
2. Can be used with Synplicity HDL Analyst tool

Synthesis Options

Default property values are used for the synthesis process unless you modify them. You can set properties for the Synthesize process in the Options dialog box.

To set synthesis options:

1. **From the Libero Tools menu, select *Options*.**
2. **Select Synthesis tab.**



Location: C:\synplicity\Synplify_71A\bin\synplify.exe

Additional Parameters:

Figure 6-1. Synthesis Options

3. **View and edit your preferences and click *OK*.** Click *Default* to revert to the default settings.

Synthesizing your Design

For HDL and mixed schematic-HDL designs, Libero IDE starts and loads Synplify with the appropriate design files.

To synthesize your design:

1. In Libero IDE, do one of the following:

- Right-click the HDL file in the File Manager tab and select *Synthesize* as shown in Figure 6-2.
- Right-click the top-level schematic for mixed schematic-HDL designs in the Design Hierarchy tab, and select *Synthesize* as shown in Figure 6-2.

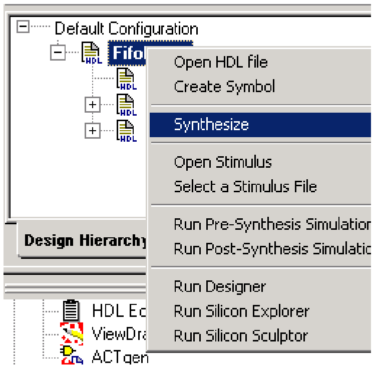
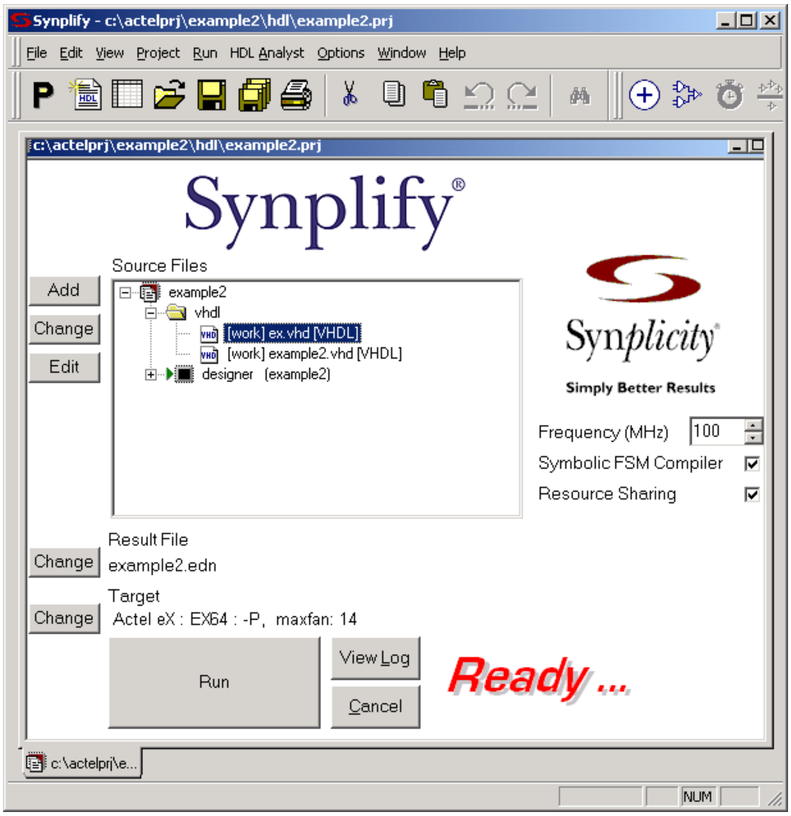


Figure 6-2. Selecting Synthesize

Synplify starts and loads the appropriate design files, with a few preset default values, as shown in Figure 6-3.


Synplify - c:\actelprj\example2\hdl\example2.prj
File Edit View Project Run HDL Analyst Options Window Help
c:\actelprj\example2\hdl\example2.prj
Synplify®
Synplicity
Simply Better Results
Add
Change
Edit
Run
View Log
Cancel
Ready ...
c:\actelprj\e...
NUM

Source Files

- example2
  - vhdl
    - [work] ex.vhd [VHDL]
    - [work] example2.vhd [VHDL]
  - designer (example2)

Frequency (MHz) 100

Symbolic FSM Compiler ✓

Resource Sharing ✓

Result File

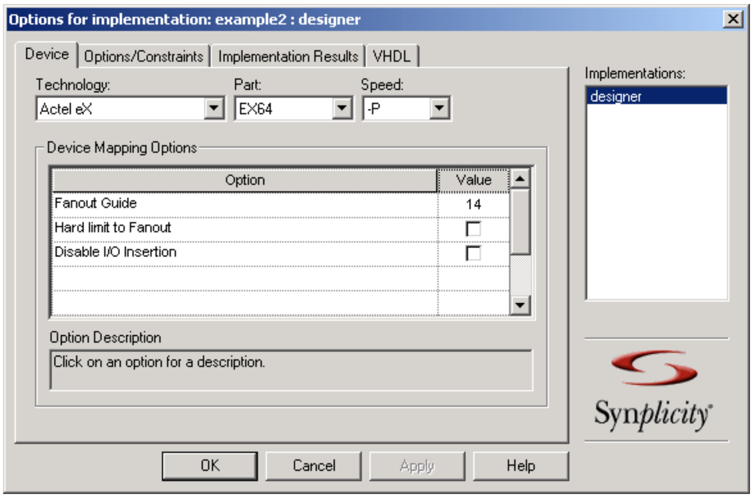
example2.edn

Target

Actel eX : EX64 : -P, maxfan: 14

Figure 6-3. Synplify Window

2. **From Synplify's Project menu, click *Implementation Options*.** This displays the Options for Implementation dialog box, as shown in Figure 6-4.



Technology: Actel eX
Part: EX64
Speed: -P

Device Mapping Options

Option	Value
Fanout Guide	14
Hard limit to Fanout	☐
Disable I/O Insertion	☐

Option Description
Click on an option for a description.

Figure 6-4. Synplify's Options for Implementation Dialog Box

3. **Set your specifications and click *OK* to dismiss the box.**

4. **Click the *RUN* button in the main window.** Synplify compiles and synthesizes the design into an EDIF (**.edn)* file. Your EDIF netlist is then automatically translated by Libero into an HDL netlist. The resulting **.edn*

and *.*v(hd)* files are visible in the File Manager, under Implementation Files, as shown in Figure 6-5

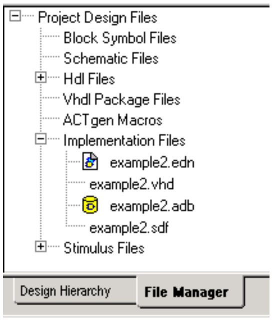


Figure 6-5. Stimulus Files in the File Manager

Note: Should any errors appear after you click the Run button, you can edit the file using the Synplify editor. Double-click the file name in the Synplify window showing the loaded design files. Any changes you make are saved to your original design file in Libero.

5. **From the File menu, click *Exit* to close Synplify.** A dialog box asks you if you would like to save any settings that you have made while in Synplify. Click *Yes*.

7 Design Implementation

This chapter describes how to implement your design using the Actel Designer software. For information on the Designer interface, please see page 149.

After you have created or imported your source files into Libero and run functional simulation, you are ready to implement your design using Designer. This involves the following steps:

1. Generating your EDIF Netlist
2. Starting Designer
3. Compiling your design
4. Using the Designer User Tools (optional)
5. Performing Layout on your design (place-and-route)
6. Back-Annotating your design
7. Generating your Programming Files

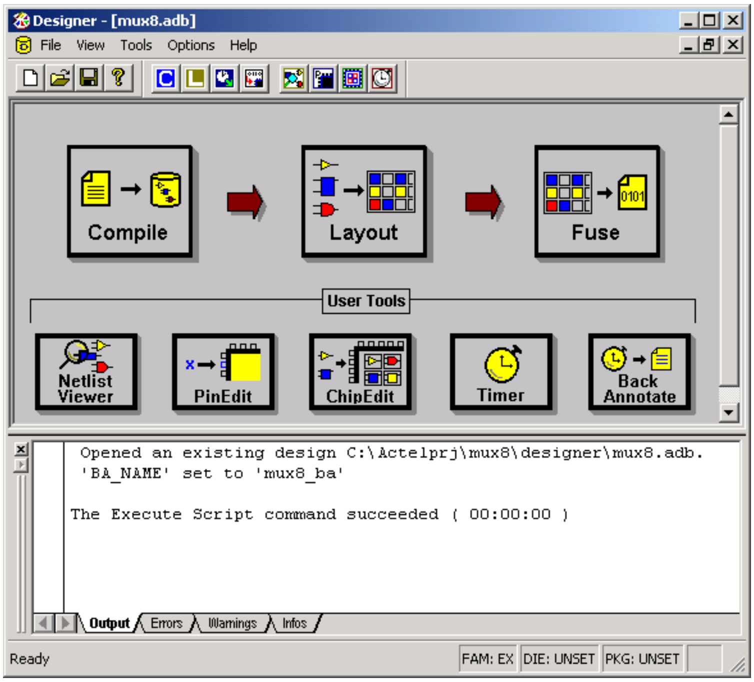
After using Designer, perform post-layout timing simulation with ModelSim for Actel.

Generating an EDIF Netlist

To generate an EDIF netlist for an HDL only or mixed schematic-HDL design, you must synthesize the HDL structural netlist in Synplify. Refer to "Synthesis" on page 103 for details.

Starting Designer

To start Designer, right-click the top level module in the Design Hierarchy and select *Run Designer*, or double click *Designer* in the Process window. Actel's Designer application starts and your design file is read in,
as shown in Figure 7-1.



```
 Opened an existing design C:\Actelprj\mux8\designer\mux8.adb.
 'BA_NAME' set to 'mux8_ba'

The Execute Script command succeeded ( 00:00:00 )
```

Figure 7-1. Designer Starts

Importing Source Files

Design implementation begins with importing source files. Libero IDE imports your netlist for you, however you can import other constraint files after Designer starts. While these files will not appear in the Libero file manager, they will remain part of your design. Source files include the files in Table 7-1:

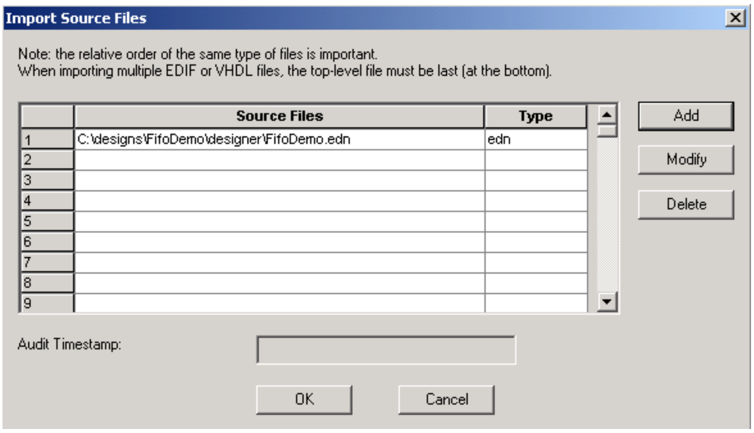
Table 7-1. Source Files

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

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

To import a source file:

1. **In Designer, from the File menu, click *Import Source Files*.** This displays the Import Source Files dialog box, as shown in Figure 7-2.



Note: the relative order of the same type of files is important.
When importing multiple EDIF or VHDL files, the top-level file must be last (at the bottom).

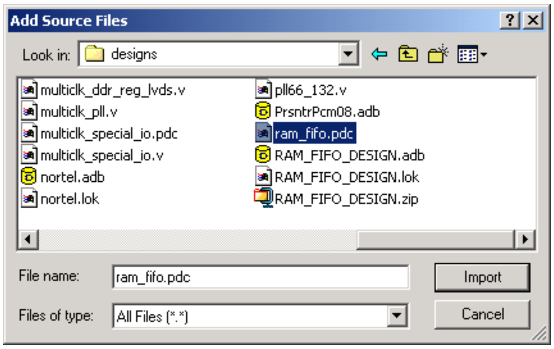
	Source Files	Type
1	C:\designs\FifoDemo\designer\FifoDemo.edn	edn
2		
3		
4		
5		
6		
7		
8		
9		

Audit Timestamp:

Figure 7-2. Import Source Files Dialog Box

Your netlist already appears because Libero imports it for you.

2. **Click the *Add* button.** The Add Source Files dialog appears, as shown in Figure 7-3.



Look in: designs

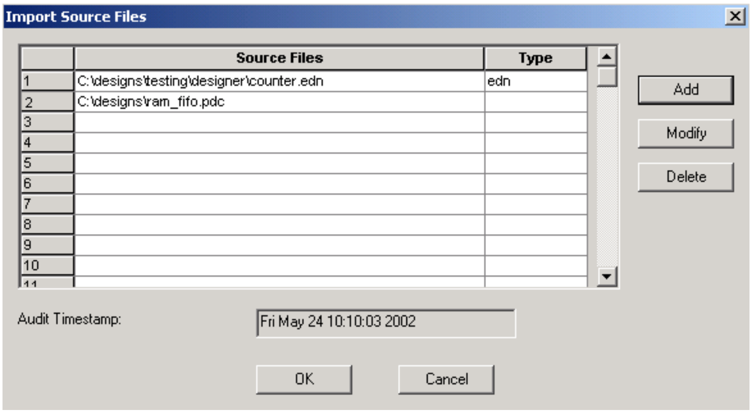
multiclk_ddr_reg_lvds.v
multiclk_pll.v
multiclk_special_io.pdc
multiclk_special_io.v
nortel.adb
nortel.lok
pll66_132.v
PrsntrPcm08.adb
ram_fifo.pdc
RAM_FIFO_DESIGN.adb
RAM_FIFO_DESIGN.lok
RAM_FIFO_DESIGN.zip

File name: ram_fifo.pdc

Files of type: All Files (*.*)

Figure 7-3. Add Source File Dialog Box

3. **Select the file you wish to add and click *Import*.** The file is added to the Import Source Files dialog box.as shown in Figure 7-4.



	Source Files	Type
1	C:\designs\testing\designer\counter.edn	edn
2	C:\designs\ram_fifo.pdc	
3		
4		
5		
6		
7		
8		
9		
10		

Audit Timestamp: Fri May 24 10:10:03 2002

Figure 7-4. Import Source Files Dialog Box with PDC File Added

4. **Add additional source files to the list.** All files added to the Import Source Files dialog box are imported at the same time.

   If you need to modify a selection, select the file row and click *Modify*.

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

5. **Ordering your source files.** Select and drag your files to specify the import order. Specifying an order is useful if you are importing multiple netlist files, .gcf files, or .pdc files. When importing multiple EDIF or structural HDL files, the top-level file must appear last in the list (at the bottom).

6. **After you are done adding all your source files, click *OK*.** Your source files are imported. Any errors appear in Designer's Log Window.

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

Importing Auxiliary Files

Auxiliary Files are listed in Table 7-2:

Table 7-2. Auxiliary Files

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

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

Note: Criticality (.crt) is a legacy file format. It is supported for the ACT-MX familes only.

To import an auxiliary file:

1. **From the File menu, click *Import Auxiliary Files.*** The Import Auxiliary Files dialog appears, as shown in Figure 7-5.

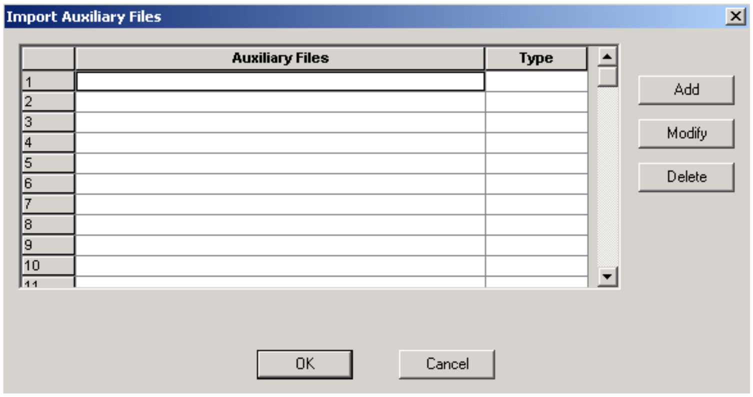
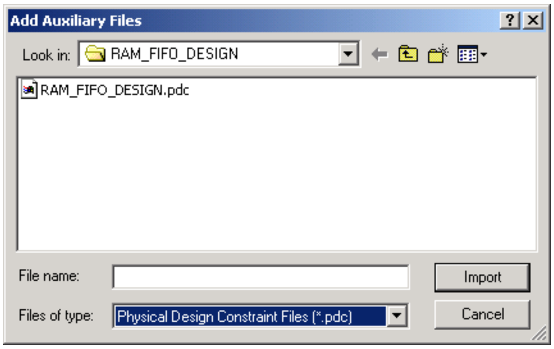


Figure 7-5. Import Auxiliary Files Dialog Box

2. **Click *the Add* button.** The Add Auxiliary Files dialog box appears, as shown in Figure 7-6.



Look in: RAM_FIFO_DESIGN

RAM_FIFO_DESIGN.pdc

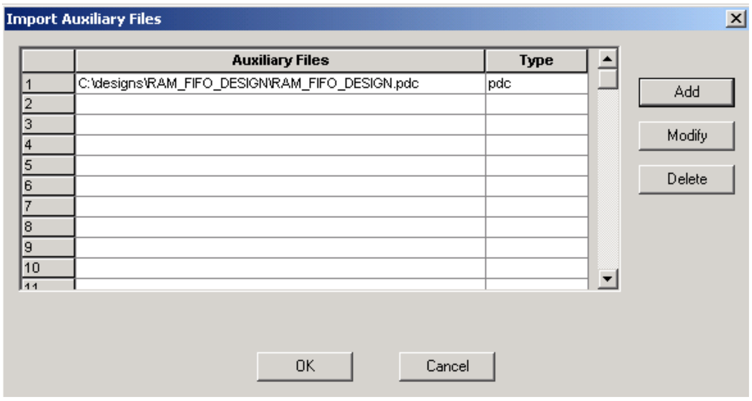
File name:

Files of type: Physical Design Constraint Files (*.pdc)

Figure 7-6. Add Auxiliary Files

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

3. **Select your file and click *Import*. The file is added to the Import Auxiliary Files dialog box, as shown in Figure 7-7.**



	Auxiliary Files	Type
1	C:\designs\RAM_FIFO_DESIGN\RAM_FIFO_DESIGN.pdc	pdc
2		
3		
4		
5		
6		
7		
8		
9		
10		

Figure 7-7. File Added to the Import Auxiliary Files Dialog

4. **Continue to add more auxiliary files to the list.**
   - To modify a selection, select the file row and click *Modify*.
   - To delete a file, select the file row and click *Delete*.
5. **Ordering your source files.** Select and drag your files to specify the import order. Specifying a priority is useful if you are importing multiple netlist files, .gcf files, or .pdc files.
6. **After you are done adding all your Auxiliary files, click *OK*.** Your auxiliary files are imported. Any errors appear in Designer's Log Window.

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

Importing PDC Files (Axcelerator family only)

Physical Design Constraint (PDC) files can specify:

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

The Axcelerator family of devices supports multiple I/O standards (with different I/O voltages) in a single die. You can use ChipEdit and PinEdit to set I/O standards and attributes, or alternatively you can export and import this information in a PDC file.

Physical Design Constraint (PDC) files are Tcl script files. For information on Tcl, see the Scripting chapter in the Designer User's Guide.

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

To import a PDC file:

1. **From the File menu, click *Import Auxiliary Files*.** The Import Auxiliary Files dialog appears, as shown in Figure 7-8.

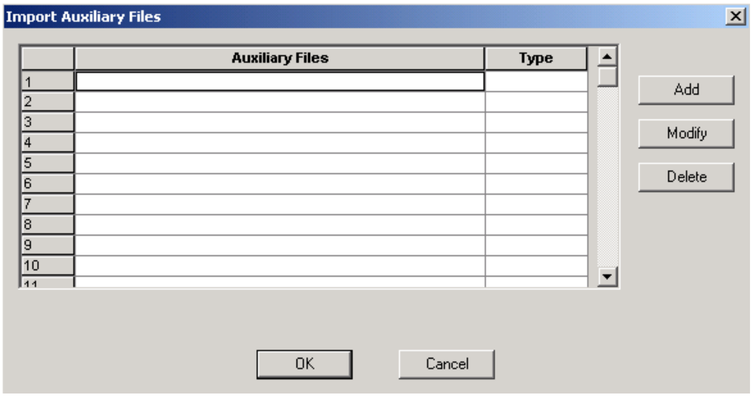
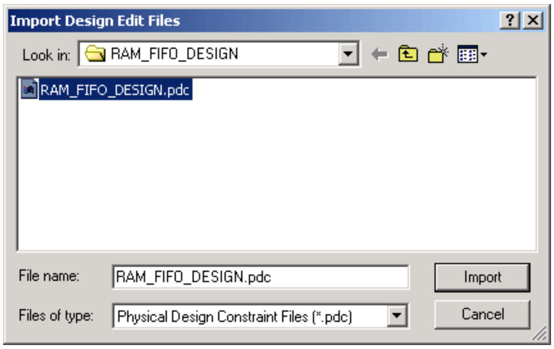


Figure 7-8. Importing a PDC File Dialog Box

2. **Click *the Add* button.** The Add Auxiliary Files dialog box appears, as shown in Figure 7-9.



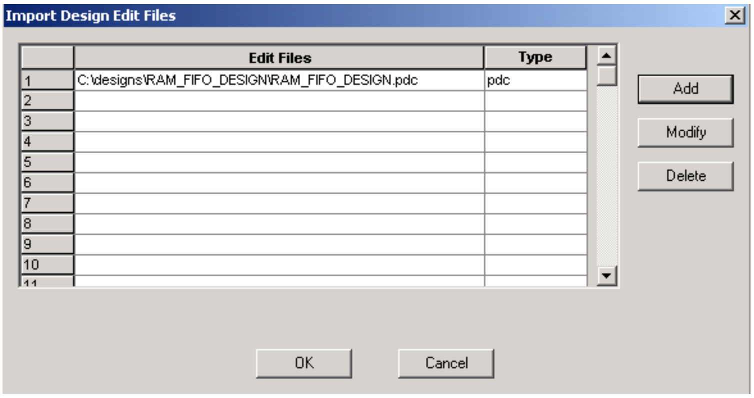
File name: RAM_FIFO_DESIGN.pdc

Files of type: Physical Design Constraint Files (*.pdc)

Figure 7-9. Selecting the PDC Files

Filter for your PDC file by selecting *Physical Design Constraint Files* (*.pdc) from the Files of Type drop-down list box.

3. **Select the PDC file and click *Import*. The file is added to the Import Auxiliary Files dialog box, as shown in Figure 7-10.**



	Edit Files	Type
1	C:\designs\RAM_FIFO_DESIGN\RAM_FIFO_DESIGN.pdc	pdc
2		
3		
4		
5		
6		
7		
8		
9		
10		

Figure 7-10. PDC File Added to the Import Auxiliary Files Dialog

4. **Click *OK*.** The PDC file is imported into Designer. Any errors appear in the Log Window.

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

Importing SDC Files

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

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

For more information on supported SDC commands and limitations, see the *Designer User's Guide.*

To import an SDC file:

1. **From the File menu, click *Import Auxiliary Files*.** The Import Auxiliary Files dialog box is displayed, as shown in Figure 7-11

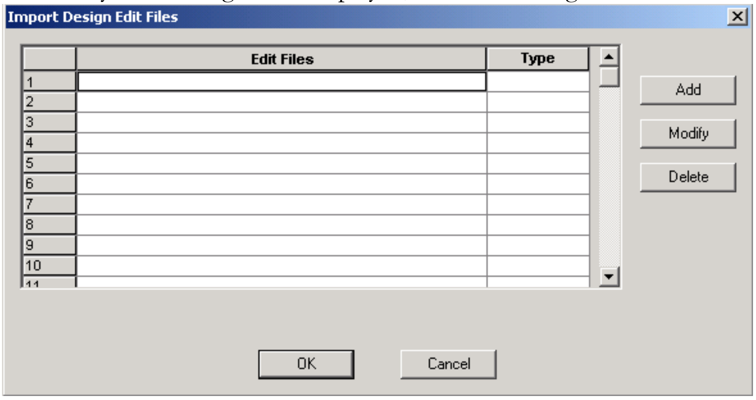


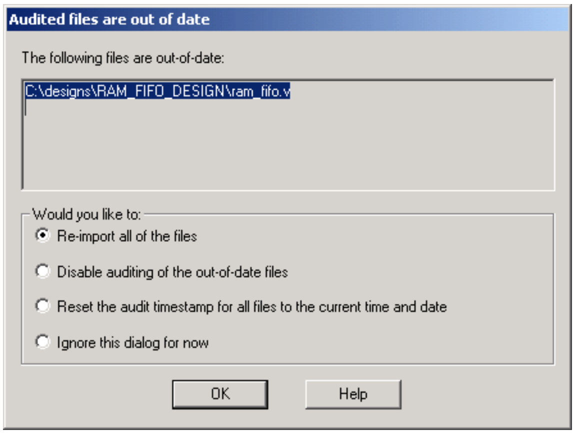
Figure 7-11. Import Auxiliary Files Dialog Box

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

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

Auditing Files

Designer audits your source files to ensure that your imported source files are current. All imported source files are date and time stamped. Designer notifies you if the file is changed, as shown in Figure 7-12.


Audited files are out of date

The following files are out-of-date:

C:\designs\RAM_FIFO_DESIGN\ram_fifo.v

Would you like to:

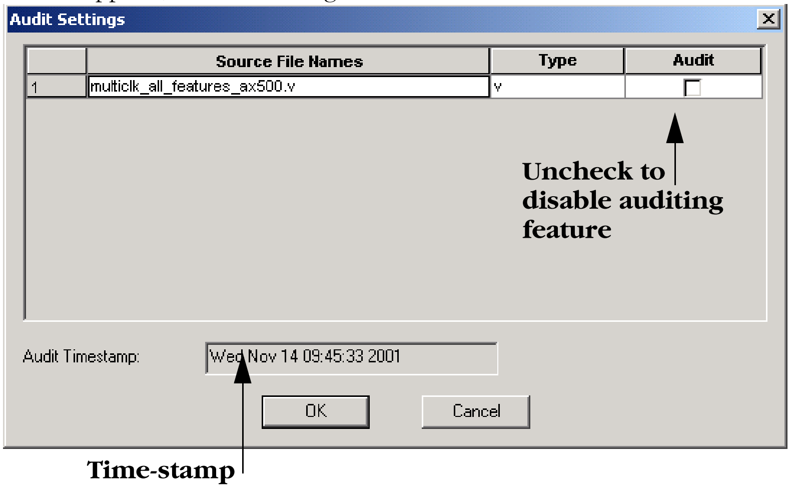
- [✓ Re-import all of the files]
- Disable auditing of the out-of-date files
- Reset the audit timestamp for all files to the current time and date
- Ignore this dialog for now

Figure 7-12. Audited File is Out of Date Dialog Box

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

To change your audit settings:

1. **From the File menu, click *Audit Settings*.** The Audit Settings dialog box appears, as shown in Figure 7-13.



Audit Settings

	Source File Names	Type	Audit
1	multiclk_all_features_ax500.v	v	☐

Audit Timestamp: Wed Nov 14 09:45:33 2001

Figure 7-13. Update Audit Source Files Dialog Box

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

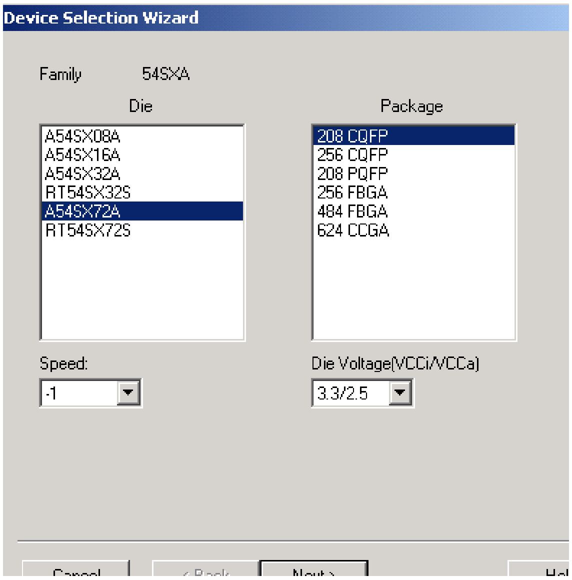
2. **Disable auditing by un-checking the audit check box next to the file.**

Device Selection Wizard

After you import your source files, the Device Selection Wizard helps you specify the device, package, and other operating conditions. (You must complete these steps before your netlist can be compiled. Starting compile without completing the device selection automatically starts the Device Selection Wizard.)

To select device, package, and other operating conditions:

1. **In the Tools menu, click *Device Selection.*** The Device Selection Wizard starts, as shown in Figure 7-14.



Family 54SXA

Speed: -1

Die Voltage(VCCi/VCCa) 3.3/2.5

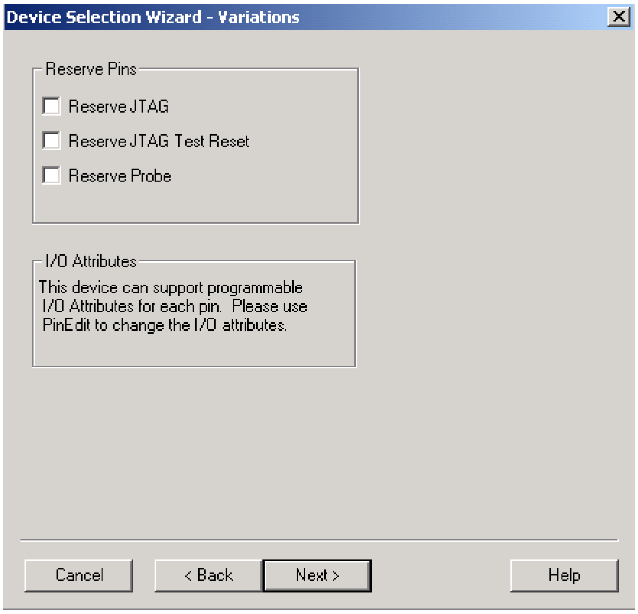
Figure 7-14. Device Selection Dialog Box

2. **Select die and package.** Select a die from the Die list. Available packages are listed for each die. Select a package.

3. **Specify speed and die voltage.** Select from the available settings in the Speed Grade and Die Voltage drop-down menus. Two numbers separated by a "/" are shown if mixed voltages are supported. If two voltages are

shown, the first number is the I/O voltage and the second number is the core (array) voltage.

4. **Click *Next*.** The Device Selection Wizard prompts you to set Variations, as shown in Figure 7-15**.**


Device Selection Wizard - Variations

Reserve Pins
- [ ] Reserve JTAG
- [ ] Reserve JTAG Test Reset
- [ ] Reserve Probe

I/O Attributes
This device can support programmable I/O Attributes for each pin. Please use PinEdit to change the I/O attributes.

Figure 7-15. Device Selection Wizard, Device Variations (Screen Varies Depending Upon Device)

5. **Set device variations.**

Note: Reserve Pins are not selectable for the Axcelerator, ProASIC, and ProASIC $^{\underline{\text{Plus}}}$ families.

Reserve Pins:

- Check the Reserve JTAG box to reserve the JTAG pins “TDI,” “TMS,” “TCK,” and “TDO” during layout.
- Check the Reserve JTAG Reset box to reserve the JTAG reset Pin “TRST” during layout.

- Check the Reserve Probe box to reserve the Probe pins "PRA," "PRB," "SDI," and "DCLK" during layout.

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, as shown in Figure 7-16.

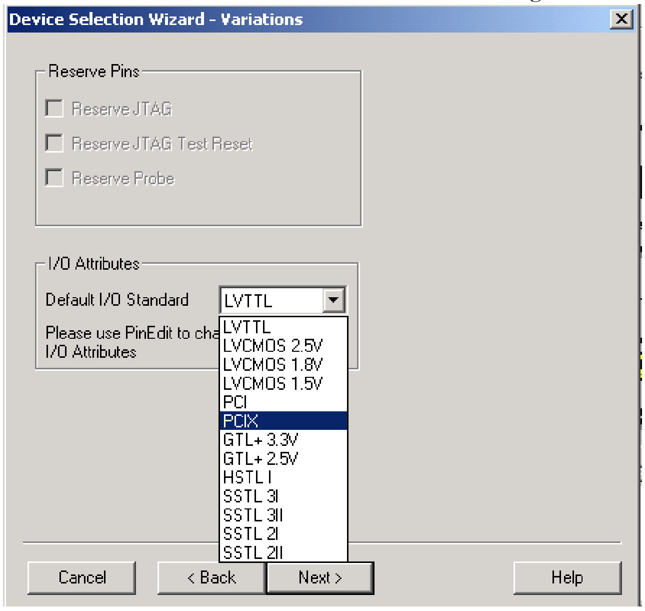
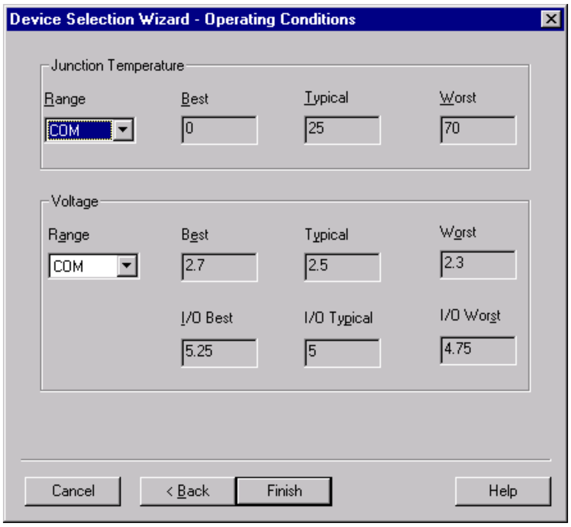


Figure 7-16. Device Selection Wizard - Variations for the Axcelerator Family

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

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



Device Selection Wizard - Operating Conditions

Junction Temperature

Range	Best	Typical	Worst
COM	0	25	70

Voltage

Range	Best	Typical	Worst
COM	2.7	2.5	2.3
	I/O Best	I/O Typical	I/O Worst
	5.25	5	4.75

Figure 7-17. Device Selection Wizard, Operating Conditions

7. **Set Operating Conditions and Click *Finish*.** Use the Operating Conditions dialog box (see Figure 7-17) to define the voltage and temperature ranges a device encounters in a working system. Supported ranges include standard industry temperature and voltage ranges, including commercial (COM), industrial (IND), and military (MIL). This displays supported ranges. Select Custom in the pull-down menu to specify a custom range. The operating condition range entered in the Operating Conditions dialog box is used by Timer, the timing report, and the back-annotation function. These tools enable you to analyze worst, typical, and best case timing. The operating conditions are summarized in Table 7-3.

Table 7-3. Operating Conditions

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

The temperature range represents the junction temperature of the device. For commercial and industrial devices, the junction temperature is a function of ambient temperature, air flow, and power consumption. For military devices, the junction temperature is a function of the case temperature, air flow, and power consumption. Because Actel devices are CMOS, power consumption must be calculated for each design. For most low power applications (e.g. 250mW), the default conditions should be adequate. You can calculate junction temperature from values in the Actel *Data Sheet*, available at http://www.actel.com/techdocs/ds/index.html. Performance decreases approximately 2.5% for every 10 degrees C that the temperature values increase. For Axcelerator, ProASIC, and the ProASIC$^{\underline{\text{PLUS}}}$ families please use SmartPower for more accurate power consumption estimation. Refer to the *SmartPower User's Guide* for more information about power consumption.

Temperature Range

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

Voltage Range

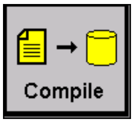
Select a voltage range from the pull-down menu (COM, IND, MIL or Custom). If you select Custom, edit the Best, Typical, and Worst fields. Modify the range to the desired value (real) such that Best $\geq$ Typical $\geq$ Worst. The top row indicates core (array) voltage, or both core and array voltages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

Compiling a Design

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

There are three ways to select the compile command:

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



- Click the *compile* icon in the toolbar.

If you have not already done so, Designer's Device Selection Wizard prompts you to set the device and package. See "Device Selection Wizard" on page 127 for information on the Device Selection Wizard.

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

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

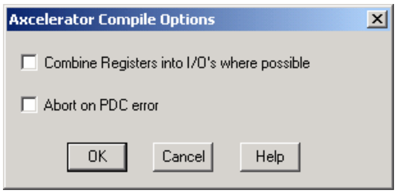
After you compile the design, you can run Layout to place-and-route the design or use the User Tools (PinEdit, ChipEdit, ProASIC Layout Viewer, Timer, SmartPower, or Netlist Viewer) to perform additional optimization prior to place-and-route.

Compile Options

The compile options are specific to each family. Compile options are not available for the ProASIC and ProASIC $\underline{PLUS}$ families.

To set compile options:

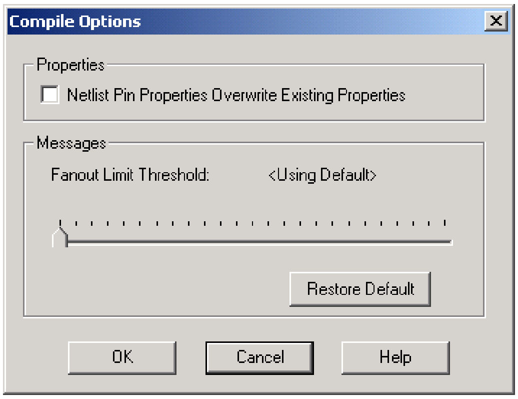
1. **From the Options menu, click *Compile*.** The compile option dialog box opens, as shown in Figure 7-18 and as shown in Figure 7-19. Options available from this dialog box are dependent upon your family.
2. **Select your options and click *OK*.** Options are explained below.


Axcelerator Compile Options

☐ Combine Registers into I/O's where possible

☐ Abort on PDC error

Figure 7-18. Compile Options Dialog Box, Axcelerator Family


Compile Options

Properties

☐ Netlist Pin Properties Overwrite Existing Properties

Messages

Fanout Limit Threshold: <Using Default>

Figure 7-19. Compile Options Dialog Box, SX Family

Netlist Pin Properties Overwrite Existing Properties

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

- If this option is off, or unchecked, then Designer uses the assignment made in PinEdit and the assignment in the netlist file for the conflicting pin is ignored.
- If this option is on, or checked, then Designer uses the assignment in the netlist file for that pin and the PinEdit assignment is ignored.

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

Combine Registers into I/Os

The Axcelerator family includes an optional register on the input path, an optional register on the output path, and an optional register on the 3-state control pin.

Select the option *Combine Registers into I/Os where possible* to take advantage of these registers.

Abort on PDC Error

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

Fanout Messages

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

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

Note: This option is available for non Axcelerator, ProASIC and ProASIC $_{\text{PLUS}}$.

User Tools

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

PinEdit

Use PinEdit to customize I/O assignments and attributes.

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

For non-Axcelerator families Actel recommends that you let Designer automatically assign I/O locations during Layout. You can then use PinEdit to optimize your design, if needed. Layout is designed to place the I/Os for optimum routability and performance. Refer to "Layout" on page 138 for information about automatically assigning I/O locations during place-and-route.

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

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

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

ChipEdit

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

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

ChipView (ProASIC and ProASIC$^{\underline{PLUS}}$ families only)

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

Timer

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

Netlist Viewer

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

SmartPower

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

Back-Annotate

The backannotation function is used to extract timing delays from your post layout data. These extracted delays are put into a file to be used by your CAE packages timing simulator. Refer to "Back-Annotate your design" on page 147 of this User's Guide for more information.

Place-and-Route Variables (Antifuse Families)

Use the Set Variable dialog box to set a variable or use Actel scripting to set variables. (For information in scripting commands, refer to the *Designer User's Guide*.) Variables must be set before running layout.

To set a variable:

1. **From the Options menu, click *Set Variable*.** The Set Variable dialog box appears as shown in Figure 7-20.

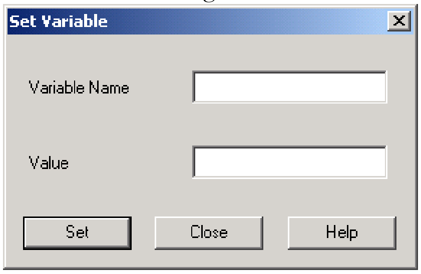


Figure 7-20. Set Variable Dialog Box

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

   PLACESEEDRANDOM

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

   PLACEACCEPTHINTS (Non-Axcelerator Families)

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

Layout

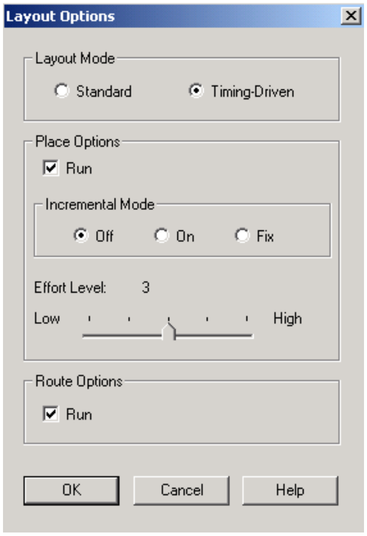
After you compile, the design is ready for layout. Layout takes the netlist information and any constraints and maps this information into the selected Actel device. Layout assigns physical locations to unassigned I/O and logic modules (placement), routing tracks to nets (routing), and calculates detailed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

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

To layout your design.

1. **There are three ways to initiate the Layout command:**
   - In the Tools menu, click *Layout*.
   - Click the Layout button in the Design Flow.
   - Click the Layout icon in the toolbar.

This displays the Layout Options dialog box, as shown in Figure 7-21.



Layout Mode
( ) Standard [✓ (•) Timing-Driven]

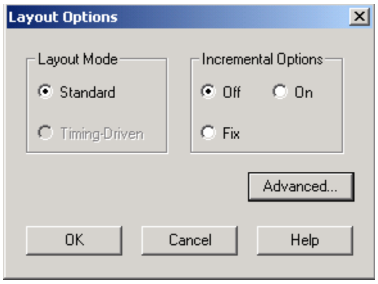
Place Options
[✓] Run

Incremental Mode
[✓ (•) Off] ( ) On ( ) Fix

Effort Level: 3
Low — High

Route Options
[✓] Run

Figure 7-21. Axcelerator Layout Options Dialog Box

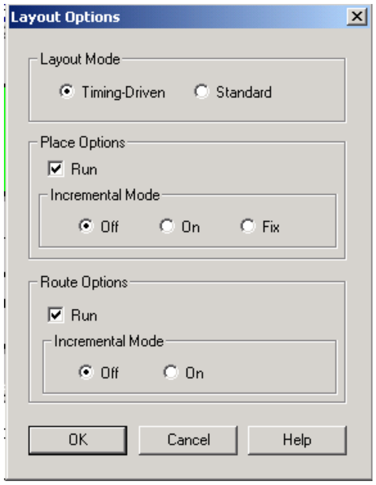


Layout Options

Layout Mode: [✓ Standard] Timing-Driven

Incremental Options: [✓ Off] On Fix

Figure 7-22. Other Anti-fuse Families Layout Options Dialog Box



Layout Options

Layout Mode: [✓ Timing-Driven] Standard

Place Options: [✓ Run]

Incremental Mode: [✓ Off] On Fix

Route Options: [✓ Run]

Incremental Mode: [✓ Off] On

Figure 7-23. ProASIC and ProASIC $^{\underline{\text{PLUS}}}$ Options Dialog Box

2. **Set your Layout options.** Options are family dependent. See "Layout Options" on page 141 for a complete description of these options.

3. **Set your Advanced Layout options (Optional).** Click the *Advanced* button in the Layout dialog box. This displays the Advanced Layout Options dialog box,as shown in Figure 7-24.

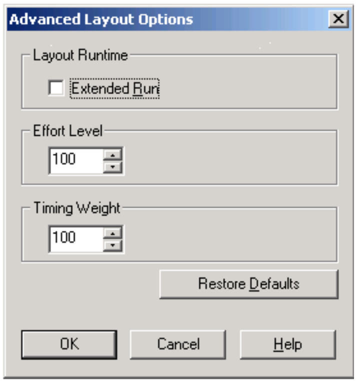


Figure 7-24. Advanced Layout Options (SX, SX-A, and eX)

Use the Advanced Layout Options dialog box to select extended run, specify an effort level, and set a timing weight.See "Layout Options" on page 141 for a complete description of these options.

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

Layout Options

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

Layout Mode

Standard

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

Standard layout does not consider delay constraints that have been set for a design during place-and-route. However, a delay report based on delay constraints entered in Timer can still be generated for the design. This is helpful to determine if Timing-Driven Layout is required.

Timing-Driven

The primary goal of Timing-Driven layout is to meet delay constraints set in Timer, SDC files (Axcelerator family only), DCF files (non-Axcelerator families), and GCF files (ProASIC and ProASIC $^{\underline{\text{PLUS}}}$) For information on GCF files, refer to the *Designer User's Guide.*

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

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

Place Options

Run

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

Incremental Placement Mode

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

In either Standard or Timing-Driven mode, the Incremental Mode option allows you to preserve the timing of a design after a successful place-and-route, even if you change part of the netlist. Incremental placement has no effect the first time you run layout. During design iteration, incremental placement attempts to preserve the placement information for any unchanged macros in a modified netlist. As a result, the timing relationships for unchanged macros approximate their initial values, decreasing the execution time to perform Layout.

By forcing Designer to retain the placement information for a portion of the design, some flexibility for optimal design layout may be lost. Therefore, do not use incremental placement to place your design in pieces. You should only use it if you have successfully run Layout and you have minor changes to your design. Incremental placement requires prior completion of Layout. Do not use incremental placement if the previous Layout failed to meet performance goals.

The FIX setting treats all unchanged macros as fixed placements. This is the strongest level of control, but it may be too restrictive for the new placement to successfully complete. The default ON setting treats unchanged macro locations as placement hints, but alters their locations as needed to successfully complete placement. Refer to the *ChipEdit User's Guide* for details on fixing macros.

For ProASIC and ProASIC $^{\underline{\text{PLUS}}}$ designs, Designer always produces a placement constraints file in the design directory called

```
<design>.dtf/last_placement.gcf.
```

This file contains all the information about the latest placement. Blocks with fixed placement constraints generate fixed placement constraints, while the others generate initial placement constraints. The existing constraint files can be edited to remove any prior placement constraints. The GCF command

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

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

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

Note: For information on .gcf files, refer to the *Designer User's Guide.*

Effort Level (Axcelerator)

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

Note: Axcelerator family only.

Route Options

Run

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

Incremental Mode

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

Incremental routing allows you to fully route a design when some nets failed to route during a previous run. You can also use it when the incoming netlist has undergone an E.C.O. (Engineering Change Order).

Incremental routing should only be used if a low number of nets fail to route (less than 50 open nets or shorted segments).

A high number of failures usually indicates a less than optimal placement (if using manual placement through macros for example) or a design that is highly connected and does not fit in the device. If a high number of nets fail, relax constraints, remove tight placement constraints, or select a bigger device and rerun routing.

Advanced Layout Options

Extended Run

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

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

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

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

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

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

Timing Weight

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

The timing weight value specifies this weight as a percentage of the default weight (i.e. a value of 100 has no effect). If you use a value less than 100, more emphasis is placed on routability and less on performance. Such a setting would be appropriate for a design that fails to route with TDPR. In case more emphasis on performance is desired, set this variable to a value higher than 100. In this case, routing failure is more likely. A very high timing value weight could also distort the optimization process and degrade performance. A value greater than 150 is not recommended.

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

Layout Failures

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

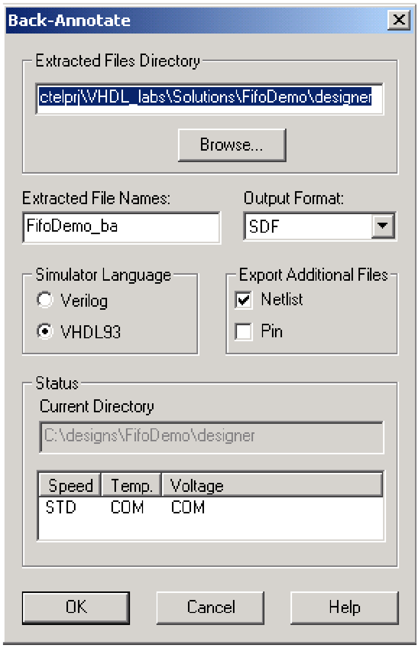
Failures During Timing Driven Layout

Layout can fail during initialization if the assigned constraints are impossible (i.e. no routing path on the device can meet the assigned constraint). You must change the circuit or relax the constraints to proceed with Timing-Driven Layout.

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

Back-Annotate your design

1. **From Designer, click *Back-Annotate*, or select *Back-Annotate* from the Tools menu.** The Back-Annotate dialog box appears, as shown in Figure 7-25.


Back-Annotate

Extracted Files Directory: ctelprj\VHDL_labs\Solutions\FifoDemo\designer

Extracted File Names: FifoDemo_ba

Output Format: SDF

Simulator Language: Verilog [✓ VHDL93]

Export Additional Files: [✓ Netlist] Pin

Status

Current Directory: C:\designs\FifoDemo\designer

Speed	Temp.	Voltage
STD	COM	COM

Figure 7-25. Back-Annotate Dialog Box

- Choose *SDF* as CAE type and appropriate simulation language.
- Select *Netlist* in the Export Additional Files area.
- 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.

As shown in Figure 7-26 below, the files counter_ba.vhd and counter_ba.sdf were exported to Libero counter.prj after Layout was run. The files are visible from the File Manager, under Implementation Files, as shown in Figure 7-26.

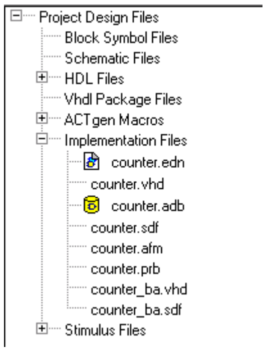

```
Project Design Files
    Block Symbol Files
    Schematic Files
    HDL Files
    Vhdl Package Files
    ACTgen Macros
    Implementation Files
        counter.edn
        counter.vhd
        counter.adb
        counter.sdf
        counter.afm
        counter.prb
        counter_ba.vhd
        counter_ba.sdf
    Stimulus Files
```

Figure 7-26. Pre-Layout Files in the File Manager

Generate a Programming File

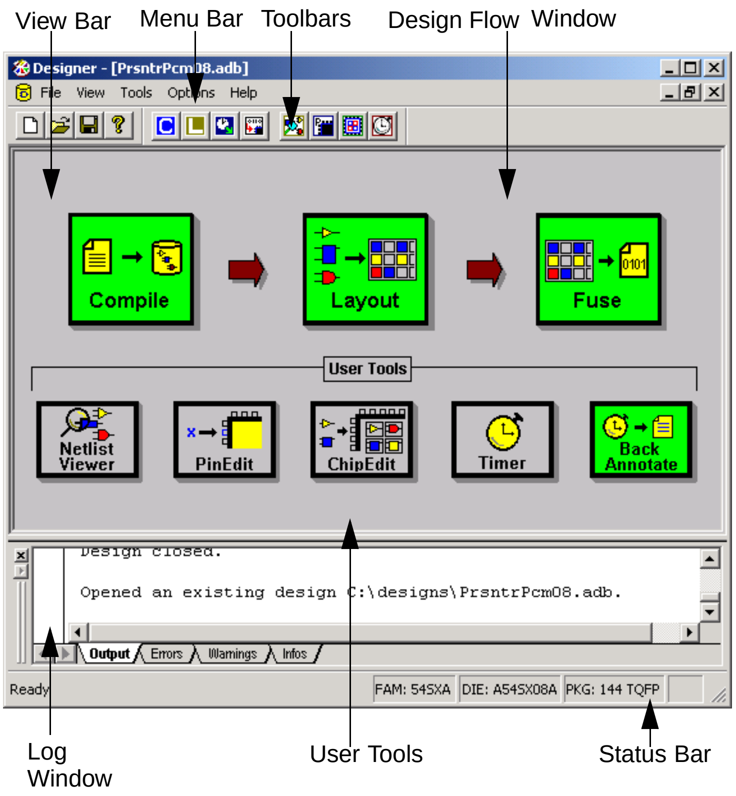
Click *Fuse* or *Bitstream* in the design flow tree if you wish to create a programming file for your design. This step can be performed later after you are satisfied with the back-annotated timing simulation. For more information, refer to “Programming” on page 213.

Save and Close Designer

From the File menu, click *Exit*. Select *Yes* to save the design before closing Designer. Designer saves all of the design information in an *.adb file. The <project>.adb file is visible in the Libero File Manger, in the Implementation Files. To re-open this file at any time, simply double-click it.

The Designer Interface

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



```
Design closed.

Opened an existing design C:\designs\PrsntrPcm08.adb.
```

Figure 7-27. Designer, Design Session Initiated (PC Version)

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

Designer Menu Commands

File Menu

New: Creates a new design.

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

Close: Closes the design.

Save: Saves the design.

Save As: Saves the design with a new name.

Execute Script: Executes a batch script.

Import Source Files: Imports netlist and constraint files.

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

Audit Settings: Changes audit settings for imported source files.

Export: Netlist: Exports the design netlist.

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

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

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

Export: Timing Files: Exports timing data for Backannotation.

Export: Script Files: Exports a script file.

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

Preferences: Sets design, internet and proxy preferences.

Recent Files: Lists recent files.

Exit: Exits Designer.

View Menu

Toolbar: Displays or hides the Toolbar.

Design Tools: Displays or hides the Design Tools toolbar.

Log Window: Displays or hides the log window.

Status Bar: Displays or hides the Status Bar.

Tools Menu

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

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

Compile: Compiles the loaded design.

Layout: Runs layout on the loaded design.

Back-Annotate: Generates delay data.

Fuse: Generates fuse data for antifuse devices.

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

PinEdit: Starts PinEdit.

ChipEdit: Starts ChipEdit.

Netlist Viewer: Starts the Netlist Viewer tool.

Timer: Starts Timer for static timing analysis.

SmartPower: Starts the power analysis tool.

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

Software Update: Checks Actel website for latest software updates.

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

Options Menu

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

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

Compile: Sets compile options specific to each family.

Get Variable: Displays a selected variable's value.

Set Variable: Sets a selected variable's value.

Clear Log: Clears the session log.

Window

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

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

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

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

Close Design: Closes the current design.

Help Menu

Help Topics: Lists help topics.

Reference Manual: Lists available online manuals.

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

License Details: Displays information about your license.

About Designer: Displays program information and current version.

Toolbar

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

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

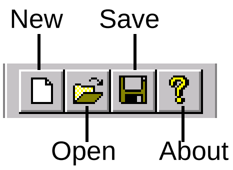


Figure 7-28. Designer Toolbar

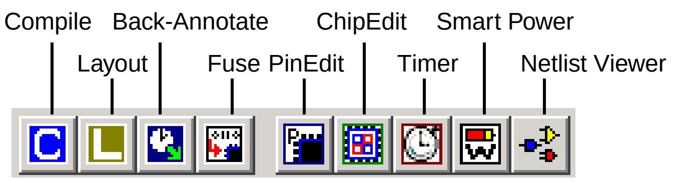


Figure 7-29. Design Tools Toolbar

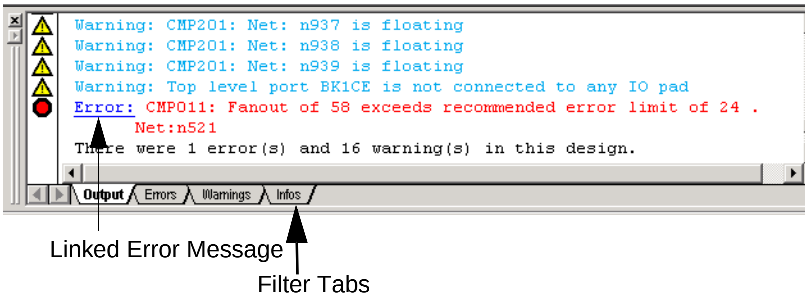
Log Window

For ProASIC and ProASIC<u>PLUS</u> families, the log window displays notes and warnings. For Antifuse families, the log window displays error, warning, and informational messages. Messages are represented by symbols and color coded:

Table 7-4.

Type	**Symbol**	**Color**
Error	●	Red
Warning	⚠	Blue
Information	●	Black

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



```
Warning: CMP201: Net: n937 is floating
Warning: CMP201: Net: n938 is floating
Warning: CMP201: Net: n939 is floating
Warning: Top level port BK1CE is not connected to any IO pad
Error: CMP011: Fanout of 58 exceeds recommended error limit of 24 .
          Net:n521
There were 1 error(s) and 16 warning(s) in this design.
```

Figure 7-30. Log Window

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

Status Bar

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

8

Creating a Test Bench

WaveFormer Lite from SynaptiCAD is the Libero IDE integrated test bench generator. This chapter describes how to create a test bench for simulation using WaveFormer Lite.

WaveFormer Lite

WaveFormer Lite is a special version of WaveFormer Pro that can generate VHDL and Verilog stimulus-based test benches for Libero IDE. WaveFormer Lite fits perfectly into the Libero design environment, automatically extracting signal information from your HDL design files, and producing HDL test bench code that can be used with any standard VHDL or Verilog simulator.

WaveFormer Lite generates VHDL and Verilog test benches from drawn waveforms. WaveFormer Lite can generate the following:

- VHDL transport test bench (*.vhd) that uses assignment statements
- VHDL wait test bench (*.vhd) that uses wait statements
- Verilog (*.v) file with Verilog stimulus statements

WaveFormer Lite Design Flow

WaveFormer Lite generates VHDL and Verilog test benches from drawn waveforms. There are four basic steps for creating test benches using WaveFormer Lite and Libero. These steps are described in detail in the following sections.

To create a test bench using WaveFormer Lite:

1. **After design creation, start WaveFormer Lite from Libero.** Your signal information is automatically imported directly from Libero.Signal information includes signal names, type, direction, and size.

2. **Draw Waveforms to describe the test bench.** "Drawing a Waveform" on page 163 describes how to use the mouse and the state buttons to draw waveforms.

3. **(Optional) Add VHDL Libraries and Use Clauses for VHDL export.** These libraries or packages can be included using the VHDL Libraries and Use Clauses dialog. From the Options menu, click the *VHDL Libraries and Use Clauses* menu item to open this dialog.

4. **Export VHDL or Verilog Test Bench. F**rom the Export menu, click *Export Timing Diagram* and choose the type of file to generate. You can generate a test bench with a top-level module that automatically hooks up the model under test to the test bench, or you can generate just a test bench model. Below is a detailed description of the two methods:

   To generate a Top-Level Model and a Test Bench model choose one of the "top-level" scripts from the save as type drop-down list box. The top-level module will instantiate the model under test and hook it up to the test bench. For this script to work the top-level module needs to be defined in the project. For Wave-Former Lite customers, the Actel Software should automatically set this option. Below is a list of top-level scripts:

   - VHDL Wait with Top Level TestBench (*.vhd)s
   - VHDL Transport with Top Level TestBench (*.vhd)s
   - Verilog with Top Level TestBench (*.v)s

   To generate a plain test bench model (which does not instantiate your model under test) then choose one of the VHDL or Verilog scripts. To simulate with the test bench model, you will need to write a top-level model that instantiates the test bench model and the model under test. This is the method used by Wave-Former Pro customers. Below is a list of VHDL and Verilog test bench generation scripts:

   - VHDL Wait (*.vhd)s
   - VHDL Transport (*.vhd)s
   - Verilog

   Note: If you added extra signals to the test bench and do not want to export those signals, then double click the signal's names to open the Signals Properties dialog and uncheck the Export check box.

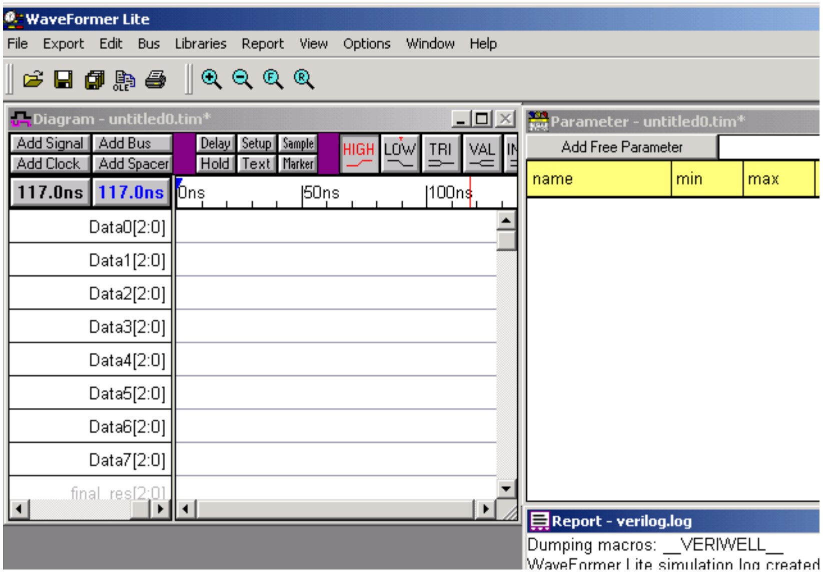
Using WaveFormer Lite

Starting WaveFormer Lite

Libero IDE automatically imports all signal information, including signal names, type, direction, and size from the top level VHDL entity or Verilog module into WaveFormer Lite when you start the program.

To start WaveFormer Lite:

1. **Double-click *WaveFormer Lite Stimulus* in the Process window.** Waveformer Lite starts, as shown in Figure 8-1.



```
Dumping macros: __VERIWELL__
WaveFormer Lite simulation log created
```

Figure 8-1. WaveFormer Lite

WaveFormer Lite consists of three windows, the Diagram, Report, and Parameter windows.

2. **Maximize the Diagram window.** The Diagram window is the main window you will be using to draw your waveforms.

Diagram Window

WaveFormer Lite's Diagram window displays the input signals you assigned to your design in the schematic, as shown in Figure 8-2.

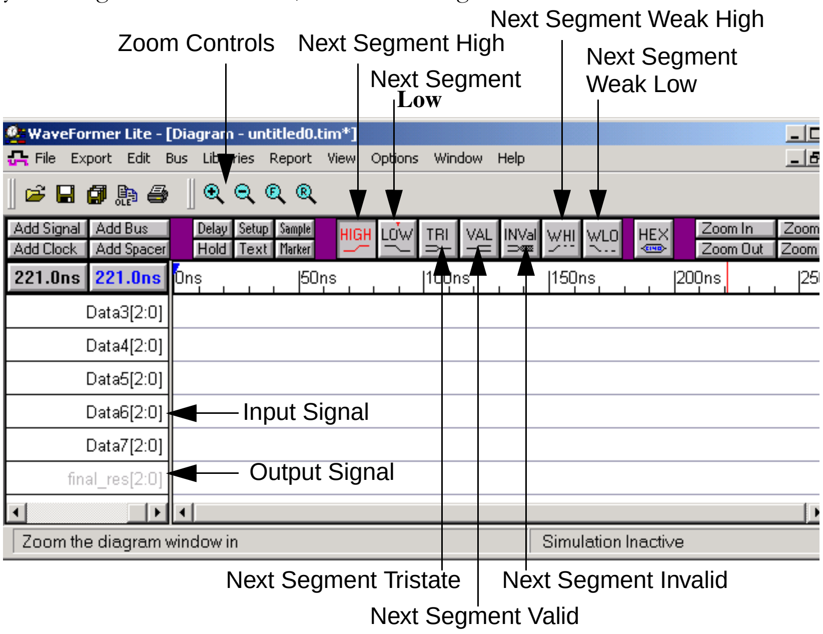


Figure 8-2. WaveFormer Lite's Diagram Window

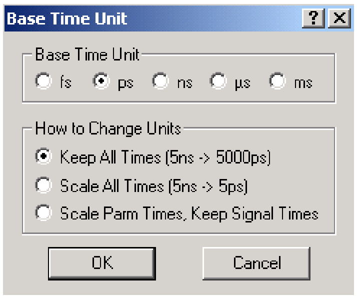
Set the Base Time Unit

Set the base time unit at the beginning of each project. The base time unit is the smallest representable amount of time that WaveFormer Lite can display. The base time unit determines the range of times that can be represented in your timing diagram. All time values are internally stored in terms of the base time unit.

Generally, it is a good idea to set the base time unit for your project one unit below the units you are working in for best rounding performance during division operations (clock frequencies are inverted and stored internally as clock periods). For example, if you were working with a circuit where the propagation times for the gates were in units of nanoseconds and the clock had a period of 20ns, you should set the base time units to picoseconds.

To set the base time unit:

1. **From the Options menu, click *Base Time Unit*.** This displays the Base Time Unit dialog box, as shown in Figure 8-3.



Base Time Unit

( ) fs [✓ ps] ( ) ns ( ) µs ( ) ms

How to Change Units

[✓ Keep All Times (5ns -> 5000ps)]
( ) Scale All Times (5ns -> 5ps)
( ) Scale Parm Times, Keep Signal Times

Figure 8-3. Base Time Unit Dialog Box

   The radio buttons set the base time unit. The other options control how any existing parameters or signals are changed when the base time unit is changed and have no effect on an empty timing diagram. (See WaveFormer Lite's on-line help for more about these options.)

2. **Select your base time unit and click *OK* to close the dialog.**

Adding and Deleting Signals

By default, you should not need to add or delete signals when using Libero IDE with WaveFormer Lite.

Setting the Display Time Unit

After setting your base time unit, you should set the display time unit. The display time unit sets the units for times which you enter and for times which are displayed. Set the display time unit to the units you most commonly use in the design.

To set the display time unit:

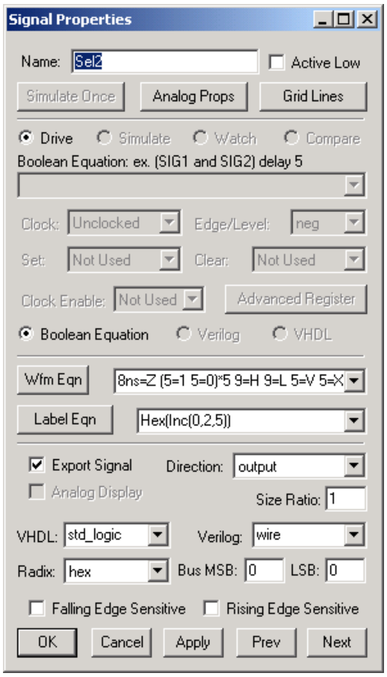
1. **From the Options menu, click *Display Unit*. This displays a sub-menu of display time units. The checked time is the current display time unit (Default is ns = nanoseconds).**

2. **Select a time unit.**

Editing Signal Properties

To edit signal properties (name, Boolean equation, export, direction, and type):

1. **Double-click the signal name to open the Signal Properties dialog box, as shown in Figure 8-4.**


Signal Properties

Name: Sel2 [ ] Active Low

Simulate Once | Analog Props | Grid Lines

(●) Drive ( ) Simulate ( ) Watch ( ) Compare

Boolean Equation: ex. (SIG1 and SIG2) delay 5

Clock: Unclocked Edge/Level: neg

Set: Not Used Clear: Not Used

Clock Enable: Not Used | Advanced Register

(●) Boolean Equation ( ) Verilog ( ) VHDL

Wfm Eqn: 8ns=Z (5=1 5=0)*5 9=H 9=L 5=V 5=X

Label Eqn: Hex(Inc(0,2,5))

[✓] Export Signal Direction: output

[ ] Analog Display Size Ratio: 1

VHDL: std_logic Verilog: wire

Radix: hex Bus MSB: 0 LSB: 0

[ ] Falling Edge Sensitive [ ] Rising Edge Sensitive

OK | Cancel | Apply | Prev | Next

Figure 8-4. Signal Properties Dialog Box

Useful properties for WaveFormer Lite include:

- **Name**: There will be a valid default name already in the edit box. The name must be at least one character long (no spaces allowed). Signal names beginning with "$$" are reserved for internal use. Signal names should not begin with a digit or contain mathematical operators. Signal names should not be the same as common Boolean operators.

- **The Export Signal check box** determines whether or not a signal will be exported to stimulus and test bench files.
- **Direction**: Used for exporting VHDL and Verilog code.
- **VHDL and Verilog**: Determine the HDL type of the signal (e.g., std_logic, integer, wire,...). The user can also enter a new type into the drop-down edit box. The VHDL and Verilog history lists can be rearranged by editing the .ini file (syncad.ini) located in the Windows directory. Be sure that WaveFormer Lite is not running when you edit the .ini file
- **Bus MSB and LSB**: Determine the bit size of the signal. A single bit signal is (0,0). This is used by the Interactive HDL Simulator to determine the result of multi-bit operations. This is also used during the export of VHDL or Verilog stimulus files and test benches to determine how to initialize the signal in the HDL model.
- **Radix**: Determines the base in which signal values are displayed.

2. **After making your selections, click *Apply* and then *OK* to dismiss the Signal Properties dialog box.**

Drawing a Waveform

After starting WaveFormer Lite, which automatically imports the signal information, the next step is to draw your waveforms in the Diagram window. Use the mouse and the state buttons to draw waveforms.

The Timing Diagram Editor is always in drawing mode.

Drawing the waveform for a regular signal

To draw the waveform of a regular signal:

1. **Select the state of the signal you wish to draw by clicking one of the state values in the toolbar.** The red state buttons determines the type of waveform drawn. The state buttons are the buttons with the waveforms drawn on their face: HIGH, LOW, TRIstate, VALid, INValid,

WHI weak high, and WLO weak low. Active state buttons are red. The active state is the type of waveform that is drawn next.

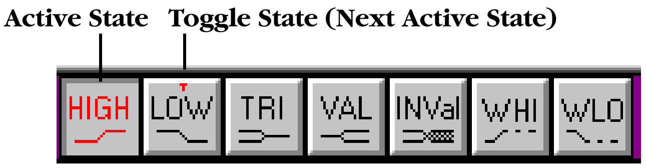


Figure 8-5. State Buttons

2. **Click a state button to activate it**. The state buttons automatically toggle between the two most recently activated states. The state with the small red “T” above the name will be the toggle state. The initial activated state is HIGH and the initial toggle state is LOW, as shown in Figure 8-5.
3. **Place the mouse cursor inside the diagram window at the same row as the signal name. Then position your mouse cursor to the time value you want the selected state to end.** The cursor shape mirrors the active red state button.
4. **Click your mouse.** This draws a waveform from the end of the previously existing signal, if any, to the mouse cursor.
5. **Move the mouse to the right and click again to draw another segment.** When you draw signals using the mouse, the signal edges are automatically aligned to the closest edge grid time. The edge grid can be controlled by selecting *Grid Settings* from the Options menu.

Drawing the Waveform for a Clock Signal

Clocks are special repetitive signals that draw themselves based on their attributes: period, frequency, duty cycle, edge jitter, offset, and other parameters. Clock edges are fixed and cannot be pushed by a delay parameter or dragged using the left mouse button. The current version of WaveFormer Lite does not allow you to associate clock properties to any automatically loaded signals.

Drawing the Waveform of a Bus Signal

WaveFormer Lite supports buses that are normal signals drawn with Valid, Invalid, and TriState states and use state information to represent bus values. The state information is added using the HEX state button.

Buses drawn with valid, invalid, and tristate states look the best. To draw a bus with only consecutive valid states click twice on the Valid state button so it stays active (state buttons do not toggle). The Valid button should be red and have a red T at the top of the button.

To edit the bus state:

1. **Double-click a segment to open the Edit Bus State dialog, or click a segment in the signal.** Then click the HEX state button on the button bar. This selects the button and opens the Edit Bus State dialog box, as shown in Figure 8-6

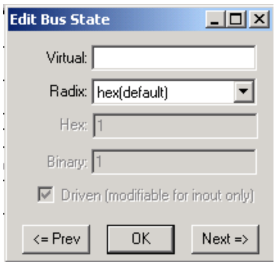


Figure 8-6. WaveFormer Lite's Edit Bus State

2. **Type the bus segment data into the Virtual field and click *OK*.**

Editing Waveforms With the Mouse

There are five mouse-based editing techniques used to modify existing waveforms. The first two techniques act on signal transitions:

To drag and drop a signal transition:

1. **Click a signal transition and drag it to the desired location.**

   Note: If you try to drag a transition past a second transition you will end up pushing the second transition unless it is fixed by a delay parameter or it is locked in the Edge Properties dialog box. When several transitions are squeezed together they look much thicker than a normal transition.

To drag-and-drop groups of signal transitions with <Shift> and

<Ctrl>:

1. **Hold down the <Shift> or <Ctrl> key while dragging a transition. This causes all the transitions to the right or left (respectively) to move with the selected edge.**

   Note: Holding down both keys causes all the edges on the entire signal to move.

The other three techniques below all act on signal segments (the waveforms between any two consecutive signal transitions). To choose the segment to operate on, left click on it. A selected segment will have a highlighted box drawn around it. If you try to select a narrow segment and one of the transitions gets selected, widen the segment by clicking the Zoom In button, which is located on the right hand corner of the button bar. To insert, change, or delete a segment:

To click-and-drag to insert a segment into a waveform:

1. **Click and hold in the middle of a wide segment (for this operation to work the segment must be wide enough to be selected).**
2. **Drag to the right and then release the mouse. A new segment will be added in the middle of the original segment.**

To change a segment's graphical state by selecting it and then pressing a state button:

1. **Click in the middle of the segment to select it.**
2. **Click a state button to apply that graphical state to the segment. State buttons are the buttons with the waveforms drawn on them.**

   Note: If you change the level of the segment to the same level as an adjacent section, the transition between them will turn red. These highlighted transitions should be deleted.

To delete a segment by selecting it and then pressing the Delete key:

1. **Click in the middle of the segment to select it.**
2. **Press the *delete* key on the keyboard.**

   Note: These techniques will only work on signals that are drawn.

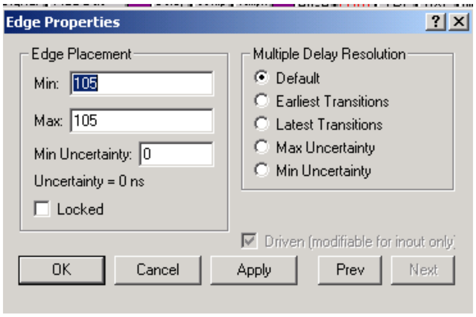
Moving and Locking Edges

Normally most waveform editing is performed using the mouse. However, edge manipulation and state values can be edited using dialogs. Dialog editing provides an interface for making precise changes to a signal's waveforms.

Moving and Locking Edges: Edges can be moved or locked using the Edge Properties dialog.

To edit a signal's edge:

1. **Double click on an edge of the signal transition to open the Edge Properties dialog, as shown in Figure 8-7.**


Edge Properties

Edge Placement
Min: 105
Max: 105
Min Uncertainty: 0
Uncertainty = 0 ns
Locked [ ]

Multiple Delay Resolution
[● Default] Earliest Transitions Latest Transitions Max Uncertainty Min Uncertainty

Driven (modifiable for inout only) [✓]

Figure 8-7. Edge Properties Dialog Box

2. **To move a signal, enter a new min or max time then click the *OK* button.**
3. **To lock an edge so that it cannot be moved, check the Locked checkbox.**

Adding a Virtual state to a Segment

In addition to the seven graphical states, a virtual data value can be added to a segment using the Edit Bus State dialog.

To add a virtual state to a segment:

1. **Double click the middle of a segment to open the Edit Bus State dialog, as shown in Figure 8-8.**

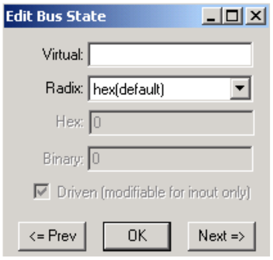


Figure 8-8. Edit Bus State Dialog Box

2. **Type a new value into the Virtual edit box. A value can be any string like "valid data", "blue", or "F3A".**

   Note: The Edit Bus State dialog is modeless so you do not have to close it to continue to do other editing functions. Each time you select a segment, that segment's information is displayed in the dialog. This makes it easy to change a lot of segments at one time. If the dialog gets in your way you can close it or roll it up by pressing the line button at the top right of the dialog window bar.

Converting Signals to Clocks

You can convert any signal to a clock.

To convert a signal to a clock:

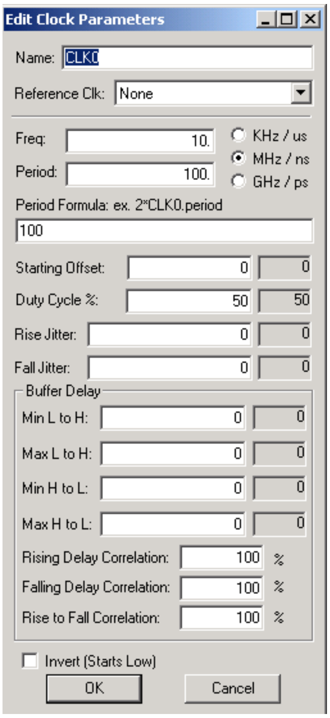
1. **Select the signal.**
2. **Right-click and select *Signal (s)* <--> *Clock(s)***

Editing Clocks

After a signal is converted to a clock (using the command *Signal(s) <-> Clock(s)*), a default setting of 10 MHz is assigned. Follow the steps below to specify the clock properties.

To edit an existing clock:

1. **Double-click a segment in the clock waveform to open the Edit Clock Parameters dialog, or double click the clock name to open up the Signal Properties dialog. Then, click the *Clock Properties* button, in the center of the dialog, to open the Edit Clock Parameters dialog.**



Edit Clock Parameters

Name: CLK0

Reference Clk: None

Freq: 10.

Period: 100.

KHz / us [✓ MHz / ns] GHz / ps

Period Formula: ex. 2*CLK0.period

100

Starting Offset:	0	0
Duty Cycle %:	50	50
Rise Jitter:	0	0
Fall Jitter:	0	0

Buffer Delay

Min L to H:	0	0
Max L to H:	0	0
Min H to L:	0	0
Max H to L:	0	0
Rising Delay Correlation:	100	%
Falling Delay Correlation:	100	%
Rise to Fall Correlation:	100	%

☐ Invert (Starts Low)

Figure 8-9. Edit Clock Parameters Dialog Box

There are three input sections in the Edit Clock Parameters dialog:

1. Label Section: There is a valid default name already in the edit box. This is

the clock signal name that was passed from Libero IDE. User generated names must be different from any other clock, signal, or bus name. The names must be at least one character (no spaces are allowed). Place the suffix $BAR at the end of a name to indicate the signal is active low. Active low signal names are displayed with a bar (line) drawn over the name. A Reference Clock lets you tightly relate the current clock to another clock in the diagram. The next section, Master Clocks and Formulas covers this feature.

2. Clock Rate Section: The clock rate can be entered as either a frequency or a period (the other will adjust itself). These must be positive real numbers whose units are controlled by the radio buttons to the right of the edit boxes. If you encounter clock frequency rounding errors, lower the Base Time Unit. The clock rate can also be set using the clock period formula box. See Master Clocks and Formulas for details.

3. Clock Properties Section:

   - Clocks are normally high at time zero unless there is a starting offset which shifts the clock transitions in time by that amount. The offset edit box accepts a valid time value or time formula.

   - Duty cycle determines how long the clock is high during a given period. Duty cycle is a percentage of the period ($0 < \text{duty} < 100$). The duty cycle edit box accepts a value between 1 and 99.

   - Edge jitter is another method of adding uncertainty. The value entered for jitter becomes an area of uncertainty both before and after the occurrence of the edge. For instance, a value of 10ns will cause a 20ns region of uncertainty centered around the selected edge.

   - Buffer delay simulates passing the clock signal through a buffer. The values represent the amount of time that the clock may or may not be valid during one of its transitions. Adding uncertainty to the clock edges causes them to be drawn with gray uncertainty regions before or after the occurrence of the edge. The buffer delay edit boxes accept valid time values or valid time formulas. Delay Correlations allow you to relate the behavior of delays within a design. The delays related to the rising and falling clocks edges can be related in the same manner using the Rising Delay Correlation, the Falling Delay Correlation and the Rise to Fall Correlation edit boxes.

   - Checking the invert box reverses the clock so that it is low at time zero (unless there is a starting offset).

Sub-Clocks and Reference Clocks

When modeling clock divider or frequency multiplier circuits, it is necessary to tightly relate the output clock to the input clock. The Reference Clock field, in the Clock Properties dialog, is used to define the relationship between the output clock (sub-clock) and the input or reference clock. This system defines how a particular edge of the reference clock will cause an edge of the sub-clock occur, and allows for accurate modeling of delay correlation, clock buffer delay removal, and edge jitter effects. To model clocks that are more loosely related use the clock formula features described in the next section.

When you choose a "reference clock" in the Clock Properties dialog, the period, offset, and duty cycle edit boxes are replaced by three new edit boxes called Divisor, Edge Offset, and Pulse Width which are all tightly related to the reference clock.

The Divisor affects the period of the new clock. A Divisor of 2 makes the new clock have twice the period of the reference clock (half the frequency, acts as a clock divider circuit). A Divisor of 0.5, divides the period by 2 (acts a frequency multiplier circuit, e.g. Phase-locked loop circuit).

Edge Offset is the number of edges of the reference clock that the new clock waits until starting. Best way to get a handle on this one is to play with it. A value of 0, starts the new clock on the rising edge of the reference clock. A value of 1, starts the new clock on the first falling edge of the reference clock. (Assuming the reference clock is not inverted). This only has meaning when frequency is being divided, multiplied clocks ignore this value.

Pulse Width is the number of edges of the reference clock that determine the width of the new clock. To get a 50% duty cycle the Pulse Width must equal the Divisor. For example a clock with a divisor of 2, has a period equal to 2 periods of the reference clock. If you count the edges on the reference clock (0,1,2) will bring you to the beginning of the second period of the reference clock. So a pulse width of 2 will give you a 50% duty cycle. This only has meaning when frequency is being divided, multiplied clocks ignore this value.

To create a sub-clock:

1. **Create a diagram with at least 2 clocks.**
2. **Double click the clock to open Clock Properties dialog.**

3. **Use the Reference Clock drop down list box to choose the reference clock name.**
4. **Adjust the Divisor, Edge Offset and Pulse Width to model your circuit.**
5. **Adjust the buffer delays and jitter values as necessary.**
   - **Click *OK* to close the dialog.**

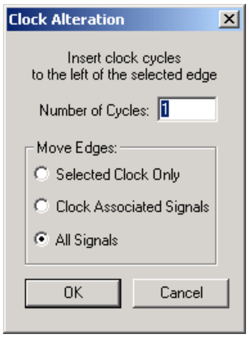
Inserting or Deleting Clock Cycles

The insertion and deletion of clock cycles operate the same way as the insert and delete keys work in a standard word processor.

To insert or delete cycles in a clock:

1. **Select a clock edge. For inserting, click the clock edge immediately to the right of where you want the cycles to be inserted. For deleting, click the first edge you want to delete.**

2. **From the Edit menu, click *Insert Clock Cycles* or *Delete Clock Cycles.*** These functions are active only when a clock edge is selected.This opens the Clock Alteration dialog, as shown in Figure 8-10.



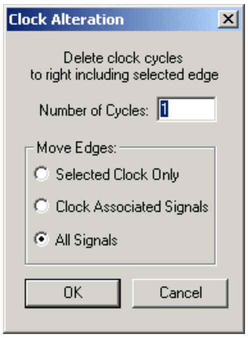
Clock Alteration

Insert clock cycles
to the left of the selected edge

Number of Cycles: 1

Move Edges:
- Selected Clock Only
- Clock Associated Signals
- [✓ All Signals]

Figure 8-10. Inserting Clock Cycles



Clock Alteration

Delete clock cycles
to right including selected edge

Number of Cycles: 1

Move Edges:
- Selected Clock Only
- Clock Associated Signals
- [✓ All Signals]

Figure 8-11. Deleting Clock Cycles

3. **Type in the number of clock cycles to perform the function on.**

4. **Move Edges.** Select which edges you wish to move:-

   - Selected Clock Only: Causes only the edges of the selected clock to move.
   - Clock Associated Signals: Causes the edges of all signals using the selected clock as their clocking signal to move.
   - All Signals: Causes the edges of all the signals in the diagram to move.

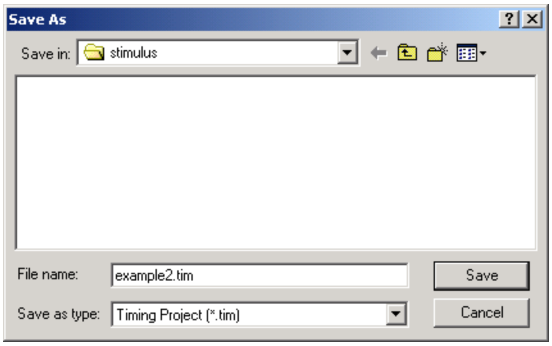
5. **Click *OK* to close the dialog.**

Saving and Exporting the HDL or Verilog Test Bench

After you are satisfied with your waveforms, you must save and export your test bench.

To export your test bench stimulus file:

1. **After creating all the waveforms successfully, select *Save As* from the File menu to save the waveforms. The Save As dialog box is displayed, as shown in Figure 8-12.**



File name: example2.tim

Save as type: Timing Project (*.tim)

Figure 8-12. WaveFormer Lite Save As Dialog Box

2. **Enter a name in the File name area and click *Save*.** Do not change the default extension of *.btim.
3. **From the Export menu, click *Export Timing Diagram As*.** Save the stimulus file as either VHDL or Verilog (according to the netlist format you exported from Designer).

   Recommended file type to export:

   - **VHDL:** Choose "VHDL Wait with Top Level TestBench (*.vhd)s"
   - **Verilog:** Choose - Verilog with Top Level TestBench (*.v)s
4. **Type a file name and click *OK*.**

The stimulus files are visible on the Libero File Manager tab, as shown in Figure 8-13.

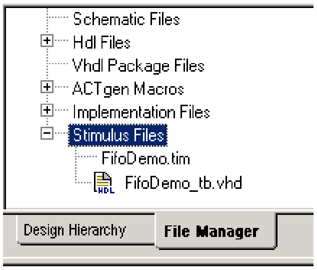


Figure 8-13. Stimulus Files are Visible in the File Manager

5. **Your design is ready to simulate using ModelSim for Actel.**

9 Simulation

This chapter describes the ModelSim for Actel interface (page 184) and the steps for performing functional (page 179) and timing simulation (page 182).

Using ModelSim for Actel

ModelSim for Actel is a custom edition of ModelSimPE that is integrated into the Libero design environment. ModelSim for Actel is an OEM edition of Model Technology Incorporated's (MTI) tools. ModelSim for Actel supports VHDL or Verilog, but it can only simulate one language at a time. It only works with Actel libraries and is supported by Actel.

Using other ModelSim Editions

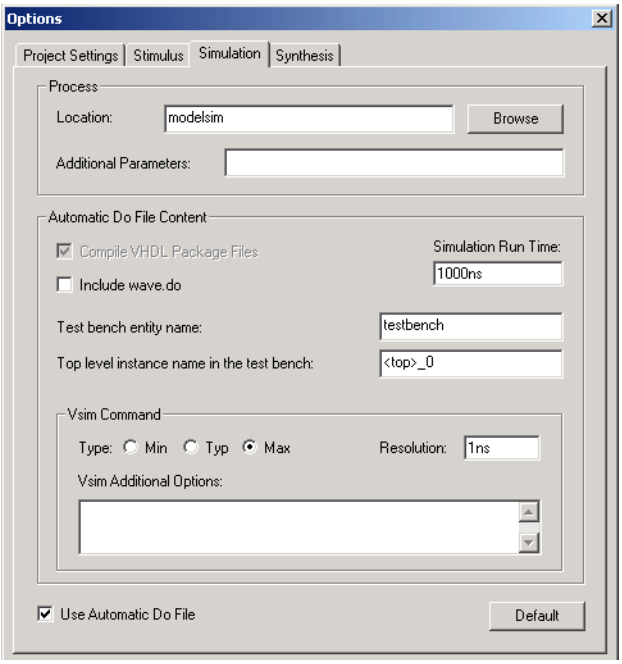
Other editions of ModelSim are supported by Libero. To use other editions of ModelSim with Libero, simply do not install ModelSim PE from the Libero CD.

Simulation Project Settings

You can set your simulation project settings in the Libero Project Manager.

To set your simulation options from Libero:

1. **From the Libero Tools menu, click *Options*.** The Options dialog box appears. Click the Simulation tab..



Options

Project Settings | Stimulus | Simulation | Synthesis

Process

Location: modelsim [Browse]

Additional Parameters:

Automatic Do File Content

[✓] Compile VHDL Package Files

[ ] Include wave.do

Simulation Run Time: 1000ns

Test bench entity name: testbench

Top level instance name in the test bench: <top>_0

Vsim Command

Type: Min Typ [✓ Max]

Resolution: 1ns

Vsim Additional Options:

[✓] Use Automatic Do File

[Default]

Figure 9-1. Setting your Simulation Options

The Simulation tab in the Options dialog box is used to specify the location of ModelSim for Actel and set your simulation preferences

- **Include wave.do:** Check this to execute the wave.do file. Use the wave.do file to customize the ModelSim Waveform window display settings.

- **Use automatic do file**: This setting creates a run.do file that is used to specify the files to be compiled and to start the simulation. By default it is checked. This setting is recommended for most users. Uncheck this box only if you do not want Libero IDE to initialize ModelSim for Actel.
- **Compile VHDL Package files**: If checked, VHDL package files are compiled in ModelSim for Actel. Default is on. Uncheck this if you do not want to compile the VHDL package files.
- **Simulation Run Time**: Specify how long the simulation should run in ns. If the value is 0, or if the field is empty, there won't be a run command included in the run.do file.
- **Test bench entity name**: Specify the name of your test bench entity name. Default is "testbench," the value used by WaveFormer Lite.
- **Top Level instance name in the test bench**: Default is <top_0>", the value used by WaveFormer Lite. Libero replaces <top> by the actual top level macro when ModelSim for Actel is run.
- **Vsim Command Type**: Select Min, Typical (Typ), or Max.
- **Resolution**: Default is 1ns.
- **Vsim additional options**: Text entered in this field is added to the vsim command.

2. **After selecting your Simulation Options, click *OK*.**

Functional Simulation

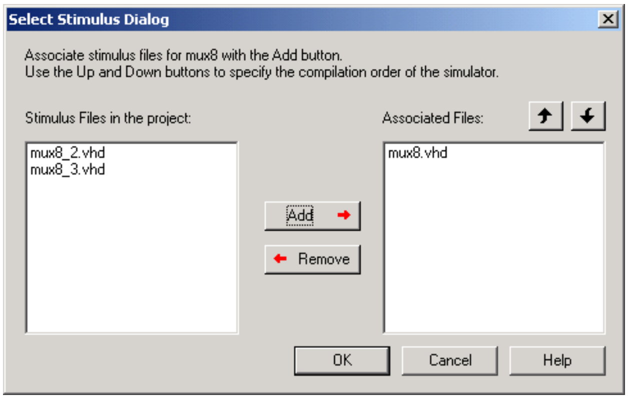
Functional simulation is performed before place-and-route and simulates only the functionality of the logic in the design.

Use the following procedure to perform a functional simulation of a design. Refer to ModelSim online help or documentation for additional information.

To perform functional simulation:

1. **Create your test bench, see** "Creating a Test Bench" on page 157**.**

2. **Select stimulus file (s).** Right-click the top level module in the Design Hierarchy Menu. Click *Select a Stimulus File* from the right-click menu. The Select Stimulus dialog box appears,as shown in Figure 9-2.



Associate stimulus files for mux8 with the Add button.
Use the Up and Down buttons to specify the compilation order of the simulator.

mux8_2.vhd
mux8_3.vhd

mux8.vhd

Figure 9-2. Select Stimulus Dialog Box

All the stimulus files in the current Libero project appear in the left Stimulus Files in the project list box. The Associated Files list box lists the test bench(es) associated with the current block.

Select the test bench you want to associated with the block in the Stimulus Files in the Project list box and click *Add* to add it to the Associated Files list.

To remove or change the file(s) in the Associated Files list box, select the file(s) and click *Remove.*

In most cases you will only have one test bench associated with your block. However, if you want simultaneous association of multiple test bench files for one simulation session, as in the case of PCI cores, add multiple files to the Select Stimulus Files dialog box. Use the up and down arrows to define the order you want the test benches compiled.

3. **When you are satisfied with the Associated Files list, click *OK*.** A check mark appears next to WaveFormer Lite in the Process window to

let you know that a test bench has been associated with the block, as shown in Figure 9-3.

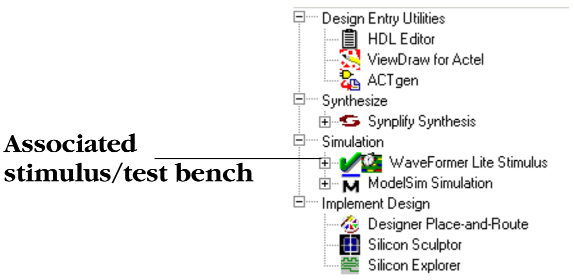
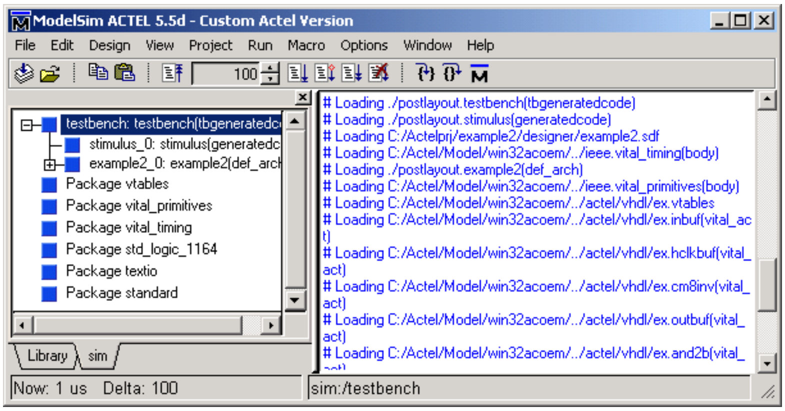


Figure 9-3. Checkmark showing Associated Test Bench with Block

4. **Start ModelSim for Actel.** There are two ways to start ModelSim for Actel.
   - Right-click the top level module in the Design Hierarchy window and select *Run Pre-Synthesis Simulation* or *Run Post-Synthesis Simulation*
   - Double-click *ModelSim Simulation* in the Process window.

   ModelSim for Actel starts and compiles the appropriate source files, as shown in Figure 9-4.


ModelSim ACTEL 5.5d - Custom Actel Version
File Edit Design View Project Run Macro Options Window Help
100
testbench: testbench(tbgeneratedc
stimulus_0: stimulus(generatedc
example2_0: example2(def_arch
Package vtables
Package vital_primitives
Package vital_timing
Package std_logic_1164
Package textio
Package standard
Library
sim
Now: 1 us Delta: 100
sim:/testbench

```
# Loading ./postlayout.testbench(tbgeneratedcode)
# Loading ./postlayout.stimulus(generatedcode)
# Loading C:/Actelprj/example2/designer/example2.sdf
# Loading C:/Actel/Model/win32acoem/../ieee.vital_timing(body)
# Loading ./postlayout.example2(def_arch)
# Loading C:/Actel/Model/win32acoem/../ieee.vital_primitives(body)
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.vtables
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.inbuf(vital_ac
t)
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.hclkbuf(vital_
act)
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.cm8inv(vital_
act)
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.outbuf(vital_
act)
# Loading C:/Actel/Model/win32acoem/../actel/vhdl/ex.and2b(vital_
```

Figure 9-4. ModelSim for Actel Compiling the Source Files

5. **When the compilation completes, the simulator runs for 1uS and the Wave window opens to display the simulation results, as shown in Figure 9-5.** (To change the 1us default runtime, see "Simulation Settings" on page 55.**)**

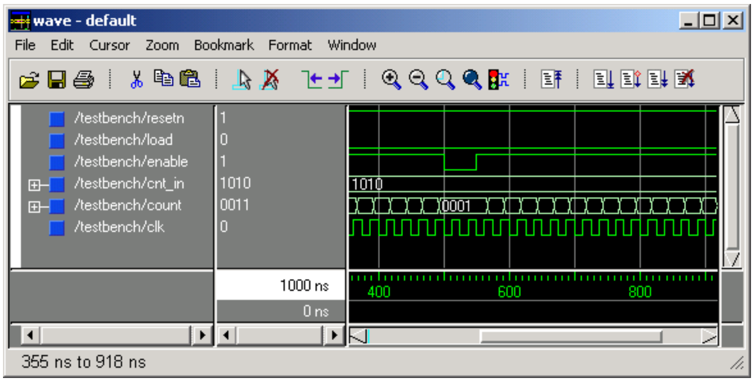


Figure 9-5. ModelSim for Actel Wave Window

6. **Scroll in the Wave window to verify that the logic of your design functions as intended.** Use the zoom buttons to zoom in and out as necessary.

7. **Exit.** From the File menu, click *Quit*.

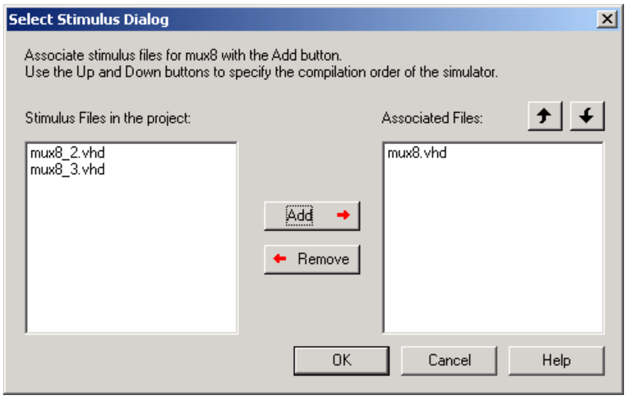
Timing Simulation

The steps for performing functional and timing simulation are nearly identical. Functional simulation is performed before place-and-route and simulates only the functionality of the logic in the design. Timing simulation is performed after the design has gone through place-and-route and uses timing information based on the delays in the placed and routed designs.

Timing simulation includes much more detailed timing information for the targeted device. Timing simulation requires a test bench (see "Creating a Test Bench" on page 157).

To perform timing simulation:

1. **If you have not done so, back-annotate your design (see page 147 for details) and create your test bench (see page 157 for details).**
2. **Select stimulus files.** Right-click the top level module in the Design Hierarchy Menu. Click *Select Stimulus* from the right-click menu. The Select Stimulus dialog box appears, as shown in Figure 9-2.


Select Stimulus Dialog

Associate stimulus files for mux8 with the Add button.
Use the Up and Down buttons to specify the compilation order of the simulator.

Stimulus Files in the project:
mux8_2.vhd
mux8_3.vhd

Associated Files:
mux8.vhd

Figure 9-6. Select Stimulus Dialog Box

All the stimulus files in the current Libero project appear in the left Stimulus Files in the project list box. The Associated Files list box lists the test bench(es) associated with the current block.

Select the test bench you want to associated with the block in the Stimulus Files in the Project list box and click *Add* to add it to the Associated Files list.

To remove or change the file(s) in the Associated Files list box, select the file(s) and click *Remove*.

In most cases you will only have one test bench associated with your block. However, if you want simultaneous association of multiple test bench files for one simulation session, as in the case of PCI cores, add multiple files to the Associated Files dialog box. Use the up and down arrows to define the order you want the test benches compiled.

3. **When you are satisfied with the Associated File(s) list, click *OK*.** A check mark appears next to WaveFormer Lite in the Process window to let you know that a test bench has been associated with the block.
4. **Start ModelSim for Actel.** Double-click *ModelSim Simulation* in the Process window. The ModelSim for Actel simulator starts and compiles the source files.When the compilation completes, the simulator runs for 1 uS and a Wave window opens to display the simulation results.
5. **Scroll in the Wave window to verify the logic works as intended.** Use the cursor and zoom buttons to zoom in and out and measure timing delays.

   Note: If you didn't create a test bench with WaveFormer Lite, you might get error messages with the vsim command if the instance names of your test bench don't follow the same conventions as WaveFormer Lite. Ignore the error message and type and the correct vsim command.
6. **Exit ModelSim for Actel.** From the File menu, click *Quit*.

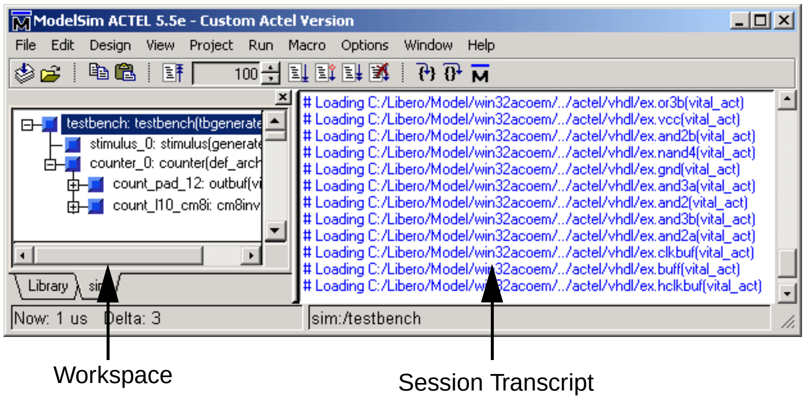
ModelSim for Actel Interface

This section describes the ModelSim for Actel graphical user interface. To find specific steps for performing functional or timing simulation, refer to "Functional Simulation" on page 179 and "Timing Simulation" on page 182.

Before you can start ModelSim for Actel, you must create a test bench (see "Creating a Test Bench" on page 157).

Main Window

The main window is the initial window that appears upon start-up. All subsequent ModelSim for Actel windows are opened from the Main window..


ModelSim ACTEL 5.5e - Custom Actel Version
File Edit Design View Project Run Macro Options Window Help
testbench: testbench(tbgenerate
stimulus_0: stimulus(generate
counter_0: counter(def_arch
count_pad_12: outbuf(vi
count_I10_cm8i: cm8inv
Library
sim
Now: 1 us Delta: 3
sim:/testbench
Workspace
Session Transcript

```
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.or3b(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.vcc(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.and2b(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.nand4(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.gnd(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.and3a(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.and2(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.and3b(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.and2a(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.clkbuf(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.buff(vital_act)
# Loading C:/Libero/Model/win32acoem/../actel/vhdl/ex.hclkbuf(vital_act)
```

Figure 9-7. ModelSim for Actel Main Window

This window contains the Workspace on the left and the Session Transcript on the right. These are described below.

Workspace

The workspace provides convenient access to projects, compiled design units, and simulation/dataset structures. It can be hidden or displayed by selecting Hide or Show Workspace from the View menu.

The Library tab shows compiled design units in the specified library. The Sim tab shows the current simulation.

Session Transcript

The session transcript contains running history of commands that are invoked and messages that occur as you work with ModelSim for Actel. When a simulation is running, the transcript displays a VSIM prompt, allowing you to enter command-line commands from within the graphic interface.

You can scroll backward and forward through the current work history by using the vertical scrollbar. You can also use arrow keys to recall previous commands, or copy and paste using the mouse within the window.

Saving the Main Window Transcript File

Variable settings determine the filename used for saving the Main window transcript. If either PrefMain(file) in modelsim.tcl, or TranscripFile in modelsim.ini file is set, then the transcript output is logged to the specified file. By default the TranscriptFile variable in modelsim.ini is set to transcript. If either variable is set, the transcript contents are always saved and no explicit saving is necessary.

To save an additional copy of the transcript with a different filename:

1. **From the File menu, click *Save Transcript As* or *Save Transcript*.** The initial save must be made with the *Save Transcript As* selection, which stores the filename in the Tcl variable PrefMain(saveFile). Subsequent saves can be made with the *Save Transcript* selection. Since no automatic saves are performed for this file, it is written only when you invoke a *Save* command. The file is written to the specified directory and records the contents of the transcript at the time of the save.

Wave Window

The Wave Window displays waveforms, and current values for the VHDL signals and variables and Verilog nets and register variables you have selected. Current and past simulations can be compared side-by-side in one Wave window.

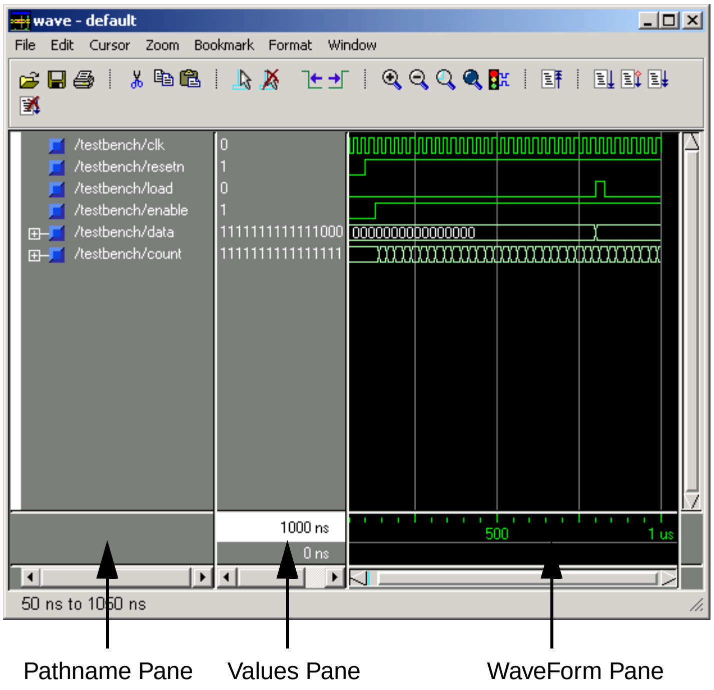


Figure 9-8. ModelSim for Actel Wave Window

Pathname Pane

The pathname pane displays signal pathnames. Signals can be displayed with full pathnames, as shown here, or with only the leaf element displayed. You can increase the size of the pane by clicking and dragging on the right border. The selected signal is highlighted.

Values Pane

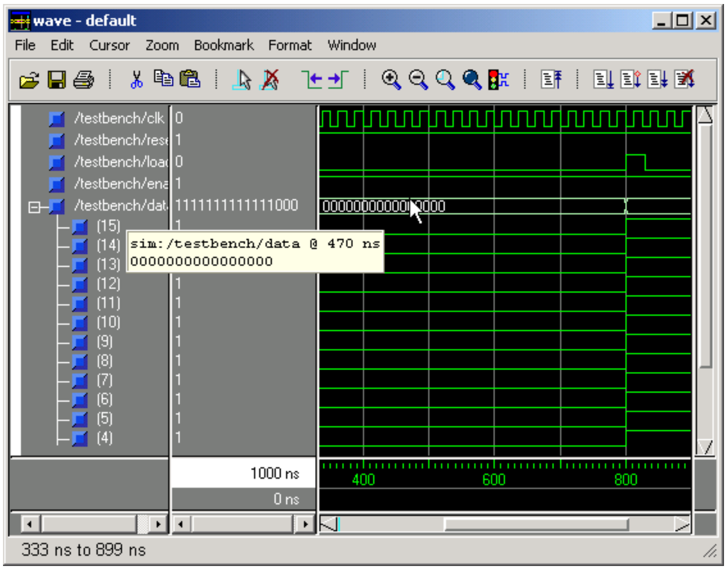
A values pane displays the values of the displayed signals. The radix for each signal can be symbolic, binary, octal, decimal, unsigned, hexadecimal, ASCII, or default. The default radix can be set by selecting Options from the Simulation (Main window)

The data in this pane is similar to that shown in the Signals window except that the values change dynamically whenever a cursor in the waveform pane (below) is moved.

Waveform Pane

The waveform pane displays the waveforms that correspond to the displayed signal pathnames. It also displays up to 20 cursors. Signal values can be displayed in analog step, analog interpolated, analog backstep, literal, logic, and event formats. Each signal can be formatted individually. The default format is logic.

If you rest your mouse pointer on a signal in the waveform pane, a popup displays with information about the signals, as shown in Figure 9-9.



```
sim:/testbench/data @ 470 ns
0000000000000000
```

Figure 9-9. Waveform Pane Popup

You can toggle this popup on and off in the Wave Window Properties dialog.

Dataflow Window

The Dataflow Window lets you trace signals and nets through your design by showing related processes.

List Window

The List Window shows the simulation values of selected VHDL signals and variables and Verilog nets and register variables in tabular format.

Process Window

The Process Window displays a list of processes in the region currently selected in the Structure window.

Signals Window

The Signals Window shows the names and current values of VHDL signals, and Verilog nets and register variables in the region currently selected in the Structure window.

Source Window

The Source Window displays the HDL source code for the design.

Structure Window

The Structure Window displays the hierarchy of structural elements such as VHDL component instances, packages, blocks, generate statements, and Verilog model instances, named blocks, tasks and functions. This same information is displayed in the Main window workspace.

Variables Window

The Variables Window displays VHDL constants, generics, variables, and Verilog register variables in the current process and their current values.

Zooming

Zooming lets you change the simulation range in the waveform display. You can zoom with either the Zoom menu, toolbar buttons, mouse, keyboard, or commands.

Using the Zoom Menu

You can use the Wave window menu bar, or call up the Zoom menu by clicking the right mouse button (of a three-button mouse) in the waveform pane.

Note: The right mouse button of a two-button mouse will not open the Zoom menu. It will, however, allow you to create a zoom area by dragging left to right while holding down the button.

The Zoom menu options include:

Zoom Full

Redraws the display to show the entire simulation from time 0 to the current simulation time.

Zoom In

Zooms in by a factor of two, increasing the resolution and decreasing the visible range horizontally.

Zoom Out

Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally.

Zoom Last

Restores the display to where it was before the last zoom operation.

Zoom Area with Mouse Button 1

Use mouse button 1 to create a zoom area. Position the mouse cursor to the left side of the desired zoom interval, press mouse button 1 and drag to the right. Release when the box has expanded to the right side of the desired zoom interval.

Zoom Range

Brings up a dialog box that allows you to enter the beginning and ending times for a range of time units to be displayed.

Zooming With the Toolbar Buttons

Use the zoom toolbar buttons as a shortcut.

Table 9-1. ModelSim for Actel's Zoom Toolbar

Zoom Button	Function
🔍	Zoom in 2x zoom in by a factor of two from the current view
🔍	Zoom out 2x zoom out by a factor of two from current view
🔍	Zoom area use the cursor to outline a zoom area
🔍	Zoom Full zoom out to view the full range of the simulation from time 0 to the current time

Zooming With the Mouse

To zoom with the mouse, position the mouse cursor to the left side of the desired zoom interval, press the middle mouse button (three-button mouse), or right button (two-button mouse), and while continuing to press, drag to the right and then release at the right side of the desired zoom interval.

Keyboard Shortcuts for Zooming

Using the following keys when the mouse cursor is within the Wave window will cause the indicated actions:

Table 9-2.

Key	Action
i I or +	zoom in
o O or -	zoom out
f or F	zoom full
l or L	zoom last
r or R	zoom range
<arrow up>	scroll waveform display up
<arrow down>	scroll waveform display down
<arrow left>	scroll waveform display left
<arrow right>	scroll waveform display right
<page up>	scroll waveform display up by page
<page down>	scroll waveform display down by page
<tab>	searches forward (right) to the next transition on the selected signal
<shift-tab>	searches backward (left) to the previous transition on the selected signal
<Control-f>	opens the find dialog box; searches within the specified field in the pathname pane for text strings

Setting Compiler Options

ModelSim for Actel's VHDL and Verilog compilers (vcom and vlog, respectively) have numerous options that affect how a design is compiled and subsequently simulated.

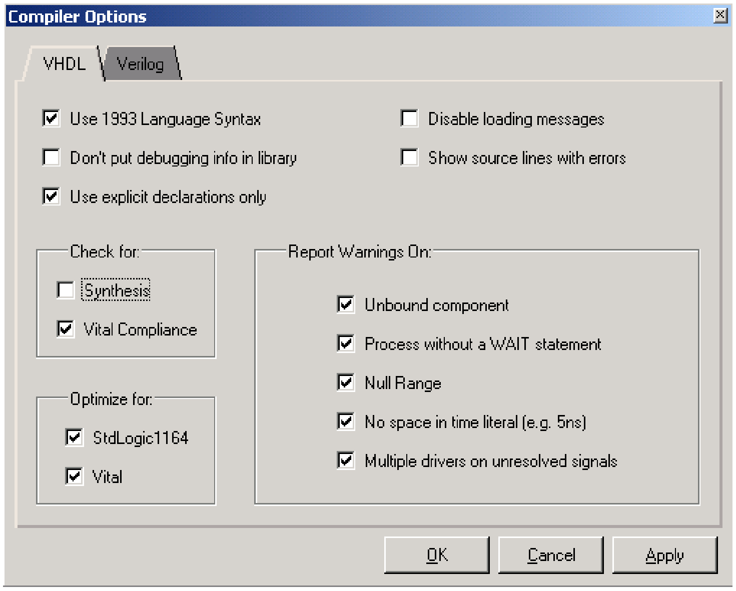
To set your compiler options:

1. **From the Options menu, click *Compile* (Main window) to bring up the Compiler Options dialog box, as shown in Figure 9-10.**

   OK accepts the changes made and closes the dialog box. *Apply* makes the changes with the dialog box open so you can test your settings. *Cancel* closes the dialog box and makes no changes. The options found on each tab of the dialog box are detailed below. Changes made in the Compiler Options dialog box become the default for all future simulations.

   Note: For information on command mentioned below, please refer to ModelSim's Command Reference Manual. From ModelSim's Help menu, click *Documentation*.

VHDL Compiler Options Tab



- [✓] Use 1993 Language Syntax
- [ ] Don't put debugging info in library
- [✓] Use explicit declarations only
- [ ] Disable loading messages
- [ ] Show source lines with errors

Check for:
- [ ] Synthesis
- [✓] Vital Compliance

Optimize for:
- [✓] StdLogic1164
- [✓] Vital

Report Warnings On:
- [✓] Unbound component
- [✓] Process without a WAIT statement
- [✓] Null Range
- [✓] No space in time literal (e.g. 5ns)
- [✓] Multiple drivers on unresolved signals

Figure 9-10. ModelSim for Actel's Compiler Options Dialog Box:VHDL

Use 1993 Language syntax

Specifies the use of VHDL93 during compilation.

Don't Put Debugging Info in Library

Models compiled with this option do not use any of the ModelSim for Actel debugging features. Consequently, your user will not be able to see into the model. This also means that you cannot set breakpoints or single step within this code. Don't compile with this option until you're done debugging. Same as the -nodebug switch for the vcom command.

Use explicit declarations only

Used to ignore an error in packages supplied by some other EDA vendors; directs the compiler to resolve ambiguous function overloading in favor of the explicit function definition. Same as the -explicit switch for the vcom command

Although it is not intuitively obvious, the = operator is overloaded in the std_logic_1164 package. All enumeration data types in VHDL get an “implicit” definition for the = operator. So while there is no explicit = operator, there is an implicit one. This implicit declaration can be hidden by an explicit declaration of = in the same package. However, if another version of the = operator is declared in a different package than that containing the enumeration declaration, and both operators become visible through use clauses, neither can be used without explicit naming, for example:

```
ARITHMETIC.”=”(left, right)
```

This option allows the explicit = operator to hide the implicit one.

Disable Loading Messages

Disables loading messages in the Main window. Same as the -quiet switch for the vcom command.

Show Source Lines with Errors

This causes the compiler to display the relevant lines of code in the transcript. Same as the -source switch for the vcom command (in the modelsim.ini file to set a permanent default.

Flag Warnings On

- **Unbound Component**:Flags any component instantiation in the VHDL source code that has no matching entity in a library that is referenced in the source code, either directly or indirectly.
- **Process without a WAIT statement**: Flags any process that does not contain a wait statement or a sensitivity list.
- **Null Range**:Flags any null range, such as 0 down to 4.
- **No space in time literal (e.g. 5ns)**: Flags any time literal that is missing a space between the number and the time unit.
- **Multiple drivers on unresolved signals**: Flags any unresolved signals that have multiple drivers.

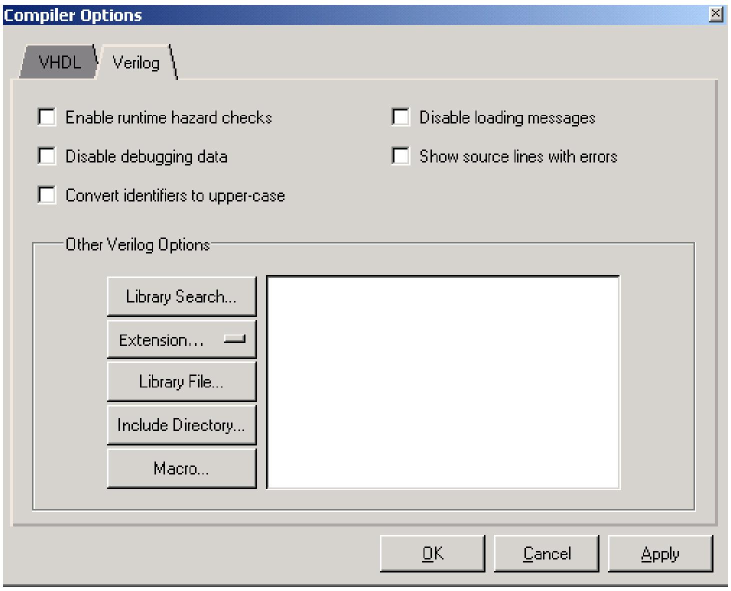
Check for

- **Synthesis**: Turns on limited synthesis-rule compliance checking.
- **Vital Compliance**: Toggle Vital compliance checking.

Optimize for

- **StdLogic1164**: Causes the compiler to perform special optimizations for speeding up simulation when the multi-value logic package std_logic_1164 is used. Unless you have modified the std_logic_1164 package, this option should always be checked.
- **Vital**: Toggle acceleration of the Vital packages.

Verilog Compiler Options



- [ ] Enable runtime hazard checks
- [ ] Disable loading messages
- [ ] Disable debugging data
- [ ] Show source lines with errors
- [ ] Convert identifiers to upper-case

Figure 9-11. ModelSim for Actel's Compiler Options Dialog Box: Verilog

Enable Run-time Hazard Checks

Enables the run-time hazard checking code. Same as the -hazards switch for the vlog command.

Disable Debugging Data

Models compiled with this option do not use any of the ModelSim for Actel debugging features. Consequently, your user will not be able to see into the model. This also means that you cannot set breakpoints or single step within this code. Don't compile with this option until you're done debugging.

Convert Verilog Identifiers to Upper-case

Converts regular Verilog identifiers to uppercase. Allows case insensitivity for module names. Same as the -u switch for the vlog command.

Disable Loading Messages

Disables loading messages in the Main window. Same as the -quiet switch for the vlog command.

Show Source Lines with Errors

Causes the compiler to display the relevant lines of code in the transcript. Same as the -source switch for the vlog command.

Other Verilog Options:

- **Library Search**: Specifies the Verilog source library directory to search for undefined modules. Same as the -y <library_directory> switch for the vlog command.

- **Extension**: Specifies the suffix of files in the library directory. Multiple suffixes can be used. Same as the +libext+<suffix> switch for the vlog command.

• **Library File:** Specifies the Verilog source library file to search for undefined modules. Same as the -v <library_file> switch for the vlog command.

- **Include Directory**: Specifies a directory for files included with the ‘include filename compiler directive. Same as the +incdir+<directory> switch for the vlog command.

- **Macro**:Defines a macro to execute during compilation. Same as the compiler directive: ‘define macro_name macro_text. Also the same as the +define+<macro_name> [=<macro_text>] switch for the vlog command.

Scripting: ModelSim´s Tcl Features

This section provides an overview of Tcl (tool command language) as used with ModelSim for Actel. Macros in ModelSim for Actel are simply Tcl scripts that contain ModelSim for Actel and, optionally, Tcl commands.

Tcl is a scripting language for controlling and extending ModelSim for Actel. Within ModelSim for Actel you can develop implementations from Tcl scripts without the use of C code. Because Tcl is interpreted, development is rapid; you can generate and execute Tcl scripts on the fly without stopping to recompile or restart ModelSim for Actel. In addition, if ModelSim for Actel does not provide the command you need, you can use Tcl to create your own commands.

Only brief descriptions are provided here; more information and command syntax can be found in these locations:

- Model*Sim's User's Guide and Command Reference Guide*. These are available from the ModelSim bookshelf (oem_docs.pdf). Look in the Libero directory, under Model*Sim*, and under Docs. Double-click on *oem_docs.pdf*
- Online Help. From Model*Sim'* Help menu, located in the Main Window, click *Tcl Syntax*, *Tcl Man Pages*, or *Tcl Help*.

Tcl Features Within ModelSim for Actel

Using Tcl with ModelSim for Actel gives you these features:

- command history (like that in C shells)
- full expression evaluation and support for all C-language operators
- a full range of math and trig functions
- support of lists and arrays
- regular expression pattern matching
- procedures
- the ability to define your own commands
- command substitution (that is, commands may be nested)
- robust scripting language for macros

Tcl References

Two books about Tcl are Tcl and the Tk Toolkit by John K. Ousterhout, published by Addison-Wesley Publishing Company, Inc., and Practical Programming in Tcl and Tk by Brent Welch published by Prentice Hall. You can also consult the following online references:

- From the Help menu in the main window, click *Tcl Man Pages*
- The Model Technology web site lists a variety of Tcl resources: www.model.com/resources/tcltk.asp

Tcl commands

More information and command syntax can be found in Model*Sim's User's Guide and Command Reference Guide*. From the Help menu (ModelSim's Main Window), click *Documentation*.

Online help is also available. From Model*Sim'* Help menu, located in the Main Window, click *Tcl Syntax*, *Tcl Man Pages*, or *Tcl Help*.

Table 9-3. Tcl ModelSim Commands

Command	Description
alias	creates a new Tcl procedure that evaluates the specified commands; used to create a user-defned alias
find	locates inerTcl classes and objects
lshift	takes a Tcl list as argument and shifts it in-place one place to the left, eliminating the 0th element
lsublist	returns a sublist of the specified Tcl list that matches the specified Tcl glob pattern
project	echoes to the Main window the current names and values of all environment variables

Model*Sim* command names that conflict with Tcl commands have been renamed or have been replaced by Tcl commands are shown in Table 9-4.

Table 9-4. New ModelSim Commands

Previous ModelSim command	Command changed to (or replaced by
continue	run with the -continue option
format list | wave	write format with either list or wave specified
if	replaced by the Tcl if command, see "if command syntax" for more information
list	add list (CR-32)
nolist | nowave	delete (CR-61) with either list or wave specified
set	replaced by the Tcl set command, see "set command syntax"
source	vsource
wave	add wave (CR-35)

Tcl Command Syntax

The following eleven rules define the syntax and semantics of the Tcl language. Additional details on if command syntax and set command syntax follow.

1. A Tcl script is a string containing one or more commands. Semi-colons and newlines are command separators unless quoted as described below. Close brackets ("]") are command terminators during command substitution (see below) unless quoted.
2. A command is evaluated in two steps. First, the Tcl interpreter breaks the command into words and performs substitutions as described below. These substitutions are performed in the same way for all commands. The first

word is used to locate a command procedure to carry out the command, then all of the words of the command are passed to the command procedure. The command procedure is free to interpret each of its words in any way it likes, such as an integer, variable name, list, or Tcl script. Different commands interpret their words differently.

3. Words of a command are separated by white space (except for newlines, which are command separators).

4. If the first character of a word is double-quote (""") then the word is terminated by the next double-quote character. If semi-colons, close brackets, or white space characters (including newlines) appear between the quotes then they are treated as ordinary characters and included in the word. Command substitution, variable substitution, and backslash substitution are performed on the characters between the quotes as described below. The double-quotes are not retained as part of the word.

5. If the first character of a word is an open brace ("{") then the word is terminated by the matching close brace ("}"). Braces nest within the word: for each additional open brace there must be an additional close brace (however, if an open brace or close brace within the word is quoted with a backslash then it is not counted in locating the matching close brace). No substitutions are performed on the characters between the braces except for backslash-newline substitutions described below, nor do semi-colons, newlines, close brackets, or white space receive any special interpretation. The word will consist of exactly the characters between the outer braces, not including the braces themselves.

6. If a word contains an open bracket ("[") then Tcl performs command substitution. To do this it invokes the Tcl interpreter recursively to process the characters following the open bracket as a Tcl script. The script may contain any number of commands and must be terminated by a close bracket ("]"). The result of the script (i.e. the result of its last command) is substituted into the word in place of the brackets and all of the characters between them. There may be any number of command substitutions in a single word. Command substitution is not performed on words enclosed in braces.

7. If a word contains a dollar-sign ("$") then Tcl performs variable substitution: the dollarsign and the following characters are replaced in the word by the value of a variable. Variable substitution may take any of the following forms:

```
$name
```

Name is the name of a scalar variable; the name is terminated by any character that isn't a letter, digit, or underscore.

```
$name(index)
```

Name gives the name of an array variable and index gives the name of an element within that array. Name must contain only letters, digits, and underscores. Command substitutions, variable substitutions, and backslash substitutions are performed on the characters of index.

```
${name}
```

Name is the name of a scalar variable. It may contain any characters whatsoever except for close braces.

There may be any number of variable substitutions in a single word. Variable substitution is not performed on words enclosed in braces.

8. If a backslash ("\") appears within a word then backslash substitution occurs. In all cases but those described below the backslash is dropped and the following character is treated as an ordinary character and included in the word. This allows characters such as double quotes, close brackets, and dollar signs to be included in words without triggering special processing. The following table lists the backslash sequences that are handled specially, along with the value that replaces each sequence.

Table 9-5.

\a	Audible alert (bell) (0x7)
\b	Backspace (0x8)
\f	Form feed (0xc)
\n	Newline (0xa)
\r	Carriage return (0xd)
\t	Tab (0x9)

Table 9-5.

\v	Vertical tab (0xb)
\<new-line>whiteSpace	A single space character replaces the backslash, newline, and all spces and tabs after the newline. This backslash sequence is unique in that it is replaced in a separate pre-pass before the command is actually parsed. This means that it will be replaced even when it occurs between braces, and the resulting space will be treated as a word separator if it isn't in braces or quotes.
\\	Backslash("\")
\ooo	The digits ooo (one, two, or three of them) give the octal value of the character
\xhh	The hexadecimal digits hh give the hexadecimal value of the character. Any number of digits may be present.

Backslash substitution is not performed on words enclosed in braces, except for backslash-newline as described above.

9. If a hash character ("#") appears at a point where Tcl is expecting the first character of the first word of a command, then the hash character and the characters that follow it, up through the next newline, are treated as a comment and ignored. The comment character only has significance when it appears at the beginning of a command.

10. Each character is processed exactly once by the Tcl interpreter as part of creating the words of a command. For example, if variable substitution occurs then no further substitutions are performed on the value of the variable; the value is inserted into the word verbatim. If command substitution occurs then the nested command is processed entirely by the recursive call to the Tcl interpreter; no substitutions are performed before making the recursive call and no additional substitutions are performed on the result of the nested script.

11. Substitutions do not affect the word boundaries of a command. For example, during variable substitution the entire value of the variable becomes part of a single word, even if the variable's value contains spaces.

If Command Syntax

The Tcl if command executes scripts conditionally. Note that in the syntax below the "?" indicates an optional argument.

Syntax

```
if expr1 ?then? body1 elseif expr2 ?then? body2 elseif ...
?else? ?bodyN?
```

Description

The **if** command evaluates *expr1* as an expression. The value of the expression must be a boolean (a numeric value, where 0 is false and anything else is true, or a string value such as **true** or **yes** for true and **false** or **no** for false); if it is true then body1 is executed by passing it to the Tcl interpreter. Otherwise *expr2* is evaluated as an expression and if it is true then body2 is executed, and so on. If none of the expressions evaluates to true then bodyN is executed. The **then** and **else** arguments are optional "noise words" to make the command easier to read. There may be any number of **elseif** clauses, including zero. BodyN may also be omitted as long as **else** is omitted too. The return value from the command is the result of the body script that was executed, or an empty string if none of the expressions was non-zero and there was no *bodyN*.

Set Command Syntax

The Tcl set command reads and writes variables. Note that in the syntax below the "?" indicates an optional argument.

Syntax

```
set varName ?value?
```

Description

Returns the value of variable *varName*. If value is specified, then sets the value of *varName* to value, creating a new variable if one doesn't already exist, and returns its value. If *varName* contains an open parenthesis and ends with a close parenthesis, then it refers to an array element: the characters before the first open parenthesis are the name of the array, and the characters between the parentheses are the index within the array. Otherwise *varName* refers to a scalar variable. Normally, *varName* is unqualified (does not include the names of any containing namespaces), and the variable of that name in the current namespace is read or written. If *varName* includes namespace qualifiers (in the array name if it refers to an array element), the variable in the specified namespace is read or written. If no procedure is active, then *varName* refers to a namespace variable (global variable if the current namespace is the global namespace). If a procedure is active, then *varName* refers to a parameter or local variable of the procedure unless the global command was invoked to declare *varName* to be global, or unless a Tcl variable command was invoked to declare *varName* to be a namespace variable.

Command Substitution

Placing a command in square brackets [ ] will cause that command to be evaluated first and

its results returned in place of the command. An example is:

```
set a 25

set b 11

set c 3

echo "the result is [expr ($a + $b)/$c]"
```

will output:

```
"the result is 12"
```

This feature allows VHDL variables and signals, and Verilog nets and registers to be accessed using:

```
[examine -<radix> name]
```

The %name substitution is no longer supported. Everywhere %name could be used, you now can use [examine -value -<radix> name] which allows the flexibility of specifying command options. The radix specification is optional.

Command Separator

A semicolon character (;) works as a separator for multiple commands on the same line. It is not required at the end of a line in a command sequence.

Multiple-line Commands

With Tcl, multiple-line commands can be used within macros and on the command line. The command line prompt will change (as in a C shell) until the multiple-line command is complete.

In the example below, note the way the opening brace '{' is at the end of the if and else lines. This is important because otherwise the Tcl scanner won't know that there is more coming in the command and will try to execute what it has up to that point, which won't be what you intend.

```
if { [exa sig_a] == "0011ZZ"} {

echo "Signal value matches"

do macro_1.do

} else {

echo "Signal value fails"

do macro_2.do }
```

Evaluation Order

An important thing to remember when using Tcl is that anything put in curly brackets {} is not evaluated immediately. This is important for if-then-else, procedures, loops, and so forth.

Tcl Relational Expression Evaluation

When you are comparing values, the following hints may be useful:

- Tcl stores all values as strings, and will convert certain strings to numeric values when appropriate. If you want a literal to be treated as a numeric value, don't quote it.

```
if {[exa var_1] == 345}...

The following will also work:

if {[exa var_1] == "345"}...
```

- However, if a literal cannot be represented as a number, you *must* quote it, or Tcl will give you an error. For instance:

  ```
  if {[exa var_2] == 001Z}...
  ```

  ```
  will give an error.
  ```

  ```
  if {[exa var_2] == "001Z"}...
  ```

  will work okay.

- Don't quote single characters in single quotes:

  ```
  if {[exa var_3] == 'X'}...
  ```

  ```
  will give an error
  ```

```
if {[exa var_3] == "X"}...
```

will work okay.

For the equal operator, you must use the C operator "==" . For not-equal, you must use the C operator "!=".

Variable Substitution

When a $<var_name> is encountered, the Tcl parser will look for variables that have been defined either by ModelSim for Actel or by you, and substitute the value of the variable.

Note: Tcl is case sensitive for variable names.

To access environment variables, use the construct:

```
$env(<var_name>)
```

```
echo My user name is $env(USER)
```

Environment variables can also be set using the env array:

```
set env(SHELL) /bin/csh
```

System Commands

To pass commands to the DOS window, use the Tcl exec command:

```
echo The date is [exec date]
```

List Processing

In Tcl a "list" is a set of strings in curly braces separated by spaces. Several Tcl commands, as shown in Table 9-7, are available for creating lists, indexing into lists, appending to lists, getting the length of lists and shifting lists.

Table 9-6. Tcl Commands for Lists

Command Syntax	Description
lappend _var_name val1 val2 ...	appends val1, val2, etc. to list var_name
lindex list_name index	returns the index-th element of the list_name; the first element is 0
linsert list_name index val1 val2 ...	inserts val1, val2, etc. just before the index-th element of list_name
list val1, val2 ...	returns a Tcl list consisting of val1, val2, etc.
llength list_name	returns the number of elements in list_name
lrange list_name first last	returns a sublist of list_name, from index first to index last; first or last may be "end," which refers to the last element in the list
lreplace list_name first last val1, val2, ...	replaces elements first through last with val1, val2, etc.

Two other commands, **lsearch** and **lsort**, are also available for list manipulation. See the Tcl man pages (From the Help menu, click *Tcl Man Pages*) for more information on these commands.

ModelSim Tcl Commands

These additional commands enhance the interface between Tcl and Model*Sim*. Only brief descriptions are provided here; more information and command syntax can be found in these locations:

- Model*Sim's User's Guide and Command Reference Guide*. From the Help menu (ModelSim's Main Window), click *Documentation*.
- Online Help. From Model*Sim'* Help menu, located in the Main Window, click *Tcl Syntax*, *Tcl Man Pages*, or *Tcl Help*.

Table 9-7. Tcl ModelSim Commands

Command	Description
alias	creates a new Tcl procedure that evaluates the specified commands; used to create a user-defned alias
find	locates inerTcl classes and objects
lshift	takes a Tcl list as argument and shifts it in-place one place to the left, eliminating the 0th element
lsublist	returns a sublist of the specified Tcl list that matches the specified Tcl glob pattern
project	echoes to the Main window the current names and values of all environment variables

ModelSim Tcl Time Commands

ModelSim Tcl time commands make simulator-time-based values available for use within other Tcl procedures. Time values may optionally contain a units specifier where the intervening space is also optional. If the space is present, the value must be quoted (e.g. 10ns, "10 ns"). Time values without units are taken to be in the UserTimeScale. Return values are always in the current Time Scale Units. All time values are converted to a 64-bit integer value in the current Time Scale. This means that values smaller than the current Time Scale will be truncated to 0.

Conversions

Table 9-8.

Command	Description
intToTime <intHi32> <intLo32>	converts two 32-bit pieces (high and low order) into a 64-bit quantity (Time in ModelSim is a 64-bit integer)
RealToTime <real>	converts a <real> number to a 64-bit integer in the current Time Scale scale
Time <time> <scaleFactor>	returns the value of <time>

Relations

Table 9-9.

Command	Description
eqTime <time> <time>	evaluates for equal
neqTime <time> <time>	evaluates for not equal
gtTime <time> <time>	evaluates for greater than
gteTime <time> <time>	evaluates for greater than or equal
ltTime <time> <time>	evaluates for less than
lteTime <time> <time>	evaluates for less than or equal

All relation operations return 1 or 0 for true or false respectively and are suitable return values for Tcl conditional expressions. For example,

```
if {[eqTime $Now 1750ns]} {

...

}
```

Arithmetic

Table 9-10. Arithmetic

addTime <time> <time>	add time
divTime <time> <time>	64-bit integer divide
mulTime <time> <time>	64-bit integer multiply
subTime <time> <time>	subtract time

10

Programming

When you are done with your design, the last step is to program your device. This chapter contains information on generating your programming files.

Generating Programming Files

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

Table 10-1. Programming Files

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

Fuse

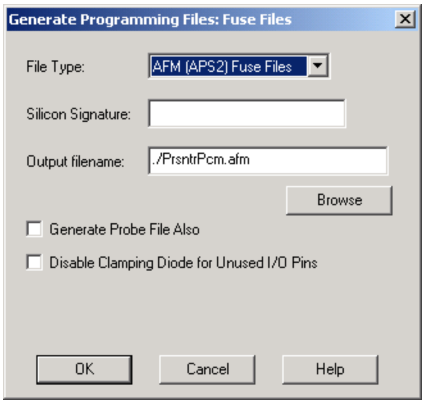
Fuse allows you to generate a programming file for your design to program an Actel device.

Silicon Signature (Antifuse Devices only)

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

To generate a programming file:

1. **In the Tools menu, click *Fuse* or click the *Fuse* button in the Design Flow window.** This displays the Generate Fuse file dialog box (Figure 10-1).



Generate Programming Files: Fuse Files

File Type: AFM (APS2) Fuse Files

Silicon Signature:

Output filename: ./PrsntrPcm.afm

Figure 10-1. Generate Fuse File Dialog Box

2. **Specify File Type.** Select the appropriate file type in the File Type pull-down menu. Select "AFM-APS2" if you are using Silicon Sculptor programmer.
3. **Enter Silicon Signature (Optional).** Enter a 5 digit hexadecimal value in the Silicon Signature box to identify the design. Valid characters are "0" through "9," and "a" through "f."
4. **Specify File name and directory in the Output filename box.** Designer automatically names the file based on the <design_name>.adb file. You can change the name by entering it in the File Name box. Click *Browse* to change the directory.

   Note: Do not add a file extension or suffix to the file name. The Designer software automatically adds the extension to the programming file name when you specify the programming format.
5. **Generate Probe File Also.** This option automatically generates a .prb file for use with Silicon Explorer

6. **Disable clamping diode for unused I/O pins. (SX-A and eX families).** Check box to disable clamping diode.
7. **Save the file.** Click *OK* when finished to save the file. The .afm file and .prb file appear in Libero's File Manager, under Implementation Files, as shown in Figure 10-2.

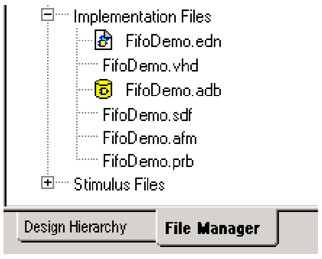


Figure 10-2. Generated Programming Files

Bitstream and STAPL Files

Bitstream allows you to generate a bitstream or STAPL file for ProASIC and ProASIC PLUS devices. Actel's ProASIC and ProASICPLUS devices contain FlashLock circuitry to lock the device by disabling the programming and readback capabilities after programming. Care has been taken to make the locking circuitry very difficult to defeat through electronic or direct physical attack.

FlashLock has three options:

No Lock

Creates a programming file which does not secure your device.

Permanent Lock

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

Keyed Lock

Within each ProASIC and ProASICPLUS device, there is a multi-bit security key user key. The number of bits depends on the size of the device. Table 10-2 and Table 10-3 show the key size of different ProASIC and ProASICPLUS devices, respectively. Once secured, read permission and write permission can only be enabled by providing the correct user key to first unlock the device.

Table 10-2. 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

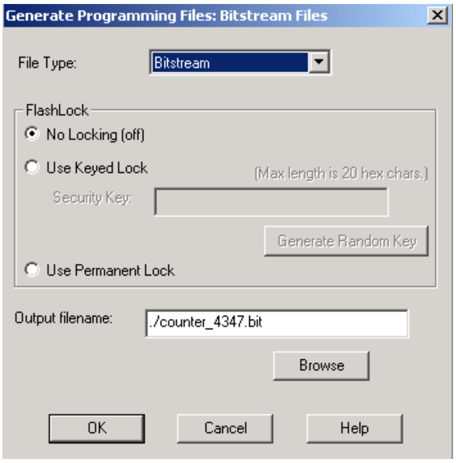
Table 10-3. Key Size of ProASIC^{PLUS} Devices

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

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

To generate a bitstream or STAPL file:

1. **In the Tools menu, click *Bitstream* or click the *Bitstream* button in the Design Flow window.** This displays the Bitstream dialog box (as shown in Figure 10-3).



File Type: Bitstream

FlashLock: (● No Locking (off)) ○ Use Keyed Lock ○ Use Permanent Lock

Security Key: (Max length is 20 hex chars.)

Output filename: ./counter_4347.bit

Figure 10-3. Bitstream Generation Dialog Box

2. **Select *Bitstream* or *STAPL* from the File Type drop-down list box.**

3. **FlashLock.** Select one of the following options:

   - No Locking: Creates a programming file which does not secure your device.
   - Use Keyed Lock: Creates a programming file which secures your device with a FlashLock key (see “Within each ProASIC and ProASICPLUS device, there is a multi-bit security key user key. The number of bits depends on the size of the device. Table 10-2 and Table 10-3 show the key size of different ProASIC and ProASICPLUS devices, respectively. Once secured, read permission and write permission can only be enabled by providing the correct user key to first unlock the device.” on page 215).
   - Use Permanent Lock: Creates a one-time programmable device.

4. **Click *OK*.** Designer validates the security key and alerts you to any concerns.

   Note: The bitstream file header contains the security key.

Programming the Security Bit

Two device programmers, Silicon Sculptor and Flash Pro, are available for ProASIC and ProASIC$^{\underline{\text{PLUS}}}$ 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$^{\underline{\text{PLUS}}}$ Flash-Based FPGAs” for more details.

Programming Tool

After you generate the programming file, you’re ready to program your device using Actel’s Silicon Sculptor programmer. An overview of this tool is provided below. For more detailed information, please refer to the *Silicon Sculptor User’s Guide*.

Silicon Sculptor II

Silicon Sculptor II is a robust, compact, single device programmer with stand alone software for the PC. Designed to allow concurrent programming of multiple units from the same PC, with speeds equivalent to, or faster than those of Actel's previous programmers. It replaces the Silicon Sculptor I as Actel's programmer of choice.

Silicon Sculptor II Benefits:

- Programs all Actel packages
- Universal Actel socket adapters
- Works with Silicon Sculptor I adapter modules
- Uses the same software as the Silicon Sculptor I
- Security fuse can be programmed to secure the devices

- Allows self-test to test its own hardware extensively

Starting the Programming Tool

To start the programming tool software:

1. **Right-click the design root file in the Design Hierarchy window.**
2. **Click *Run Silicon Sculptor*, as shown in Figure 10-4.**

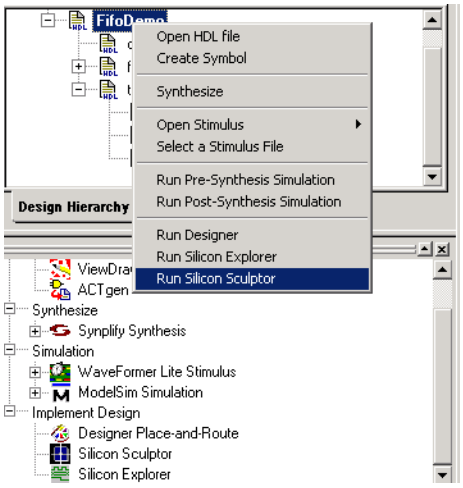


Figure 10-4. Starting Silicon Sculptor

Silicon Sculptor starts. Refer to the Silicon Sculptor User's Guide for information on using the programming tool.

A

Special Features

The special features discussed here are only applicable for pure schematic designs.

For pure schematic designs, you can include the PRESERVE and ALSPIN properties. (These are designs that do not include ACTgen macros.) Do not synthesize your design; these properties are not recognized. To generate your EDIF, see "Generating an EDIF Netlist From a Schematic" on page 223.

Actel Specific Properties in ViewDraw Schematics

PRESERVE

The Compiler in Actel's Designer Series software automatically invokes the Combiner, which attempts to combine combinatorial functions into sequential modules where ever possible. This happens in all Actel devices, except ACT 1 and 40MX which do not have S-modules. For example, you may need to have buffers in series. By default the combiner collapses the buffers into a single buffer. This "combining" does not affect the function of the circuit., but will impact delays. To prevent such combining, an ALSPRESERVE property may be added to the output net of the macro that would otherwise be combined. This property does not require a value.

To prevent this, put a "preserve" attribute on the output net of the macro that you wish to keep.

To put a PRESERVE attribute on an output net:

1. **Using ViewDraw for Actel, select the net connected to the output of the macro that you wish to preserve.**
2. **From the pull down menu select Object Attribute from the Add menu. The command line will prompt you with the following: Attribute test string:**
3. **Type *preserve* and press enter.**

ALSPIN

I/O pin assignments can be declared in the schematic or by using the pin editor (Pin Edit) in Designer. To specify I/O pin assignments in a schematic, ALSPIN attribute is placed on the nets connected to the PAD of the IO macro, as shown in Figure 10-5 and Figure 10-6.

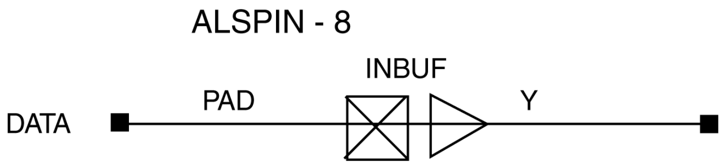


Figure 10-5. Input Buffer

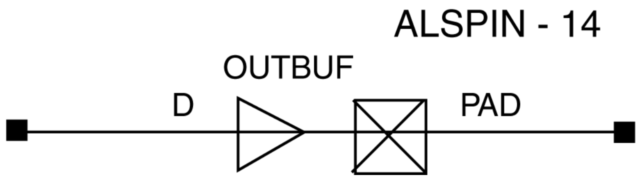


Figure 10-6. Output Buffer

In Figure 10-5, the net DATA is connected to the pad of an INBUF, and it has an associated attribute, ALSPIN, with value 8. The value 8 corresponds to the desired pin number on the target device.

To enter the ALSPIN attribute in the Net Properties dialogue box:

1. **Using ViewDraw for Actel, double-click the desired net to call up the Net Properties dialogue box.**
2. **Select the Attributes tab.**
3. **In the Name field type ALSPIN, and in the Value field type the pin number.**
4. **Click, *OK*.**

Generating an EDIF Netlist From a Schematic

While it is not part of the normal design flow, you can generate an edif netlist from a schematic and import it into your project.

To generate an edif netlist from a schematic:

1. **Right-click the schematic in the File Manager.**
2. **Select *Generate Edif* from the right-click menu, as shown in Figure A-1.**

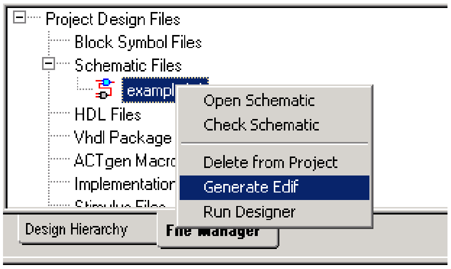


Figure A-1. Generate Edif

3. **Import the Edif into your project.** From the File menu, click Import. Select the edif file you just generated and click OK. The file appears in the File Manager under HDL files and in the Design Hierarchy.

Generating an HDL From a Schematic

While it is not part of the normal design flow, you can generate an HDL file from a schematic.

To generate an HDL file from a schematic:

1. **Select the schematic in the File Manager.**
2. **From the Process menu, select *Design Entry Utilities* and click *Generate HDL.*** The location of the file appears in the log window.

3. **Import the file into your project.** From the File menu, click *Import*. Select the file you just generated and click OK. The file appears in the File Manager under HDL files and in the Design Hierarchy.

B
Product Support

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

Actel U.S. Toll-Free Line

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

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

Customer Service

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

From Northeast and North Central U.S.A., call (408) 522-4480.
From Southeast and Southwest U.S.A., call (408) 522-4480.
From South Central U.S.A., call (408) 522-4434.
From Northwest U.S.A., call (408) 522-4434.
From Canada, call (408) 522-4480.
From Europe, call (408) 522-4252 or +44 (0) 1276 401500.
From Japan, call (408) 522-4743.
From the rest of the world, call (408) 522-4743.
Fax, from anywhere in the world (408) 522-8044.

Actel Customer Technical Support Center

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

Guru Automated Technical Support

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

Web Site

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

Contacting the Customer Technical Support Center

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

Electronic Mail

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

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

Telephone

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

(408) 522-4460
(800) 262-1060

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

Worldwide Sales Offices

Headquarters

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

Tel: 408.739.1010
Fax: 408.739.1540

US Sales Offices

California

Bay Area
Tel: 408.328.2200
Fax: 408.328.2358

Irvine
Tel: 949.727.0470
Fax: 949.727.0476

Newbury Park
Tel: 805.375.5769
Fax: 805.375.5749

Colorado

Tel: 303.420.4335
Fax: 303.420.4336

Florida

Tel: 407.977.6846
Fax: 407.977.6847

Georgia

Tel: 770.277.4980
Fax: 770.277.5896

Illinois

Tel: 847.259.1501
Fax: 847.259.1575

Massachusetts

Tel: 978.244.3800
Fax: 978.244.3820

Minnesota

Tel: 651.917.9116
Fax: 651.917.9114

New Jersey

Tel: 609.517.0304

North Carolina

Tel: 919.654.4529
Fax: 919.674.0055

Pennsylvania

Tel: 215.830.1458
Fax: 215.706.0680

Texas

Tel: 972.235.8944
Fax: 972.235.9659

International Sales Offices

Canada
235 Stafford Rd. West, Suite 106
Nepean, Ontario K2H9C1, Canada

Tel: 613.726.7575
Fax: 613.726.8666

France
361 Avenue General de Gaulle
92147 Clamart Cedex

Tel: +33 (0)1.40.83.11.00
Fax: +33 (0)1.40.94.11.04

Germany
Lohweg 27,
D-85375 Neufahrn
Germany

Tel: +49.(0)81.659.584.0
Fax: +49.(0)81.659.584.10

Italy
Via dei Garibaldini 5
20019 Settimo Milanese
Milano, Italy

Tel: +39 (0)2.3809.3259
Fax: +39 (0)2.3809.3260

Japan
EXOS Ebisu Building 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150

Tel: +81 (0)3.3445.7671
Fax: +81 (0)3.3445.7668

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

United Kingdom
Maxfli Court
Riverside Way
Camberley, Surrey
GU15 3YL
United Kingdom

Tel: +44 (0)1276.401450
Fax: +44 (0)1276.401490

Index