VIEWLOGIC

INTERFACE GUIDE

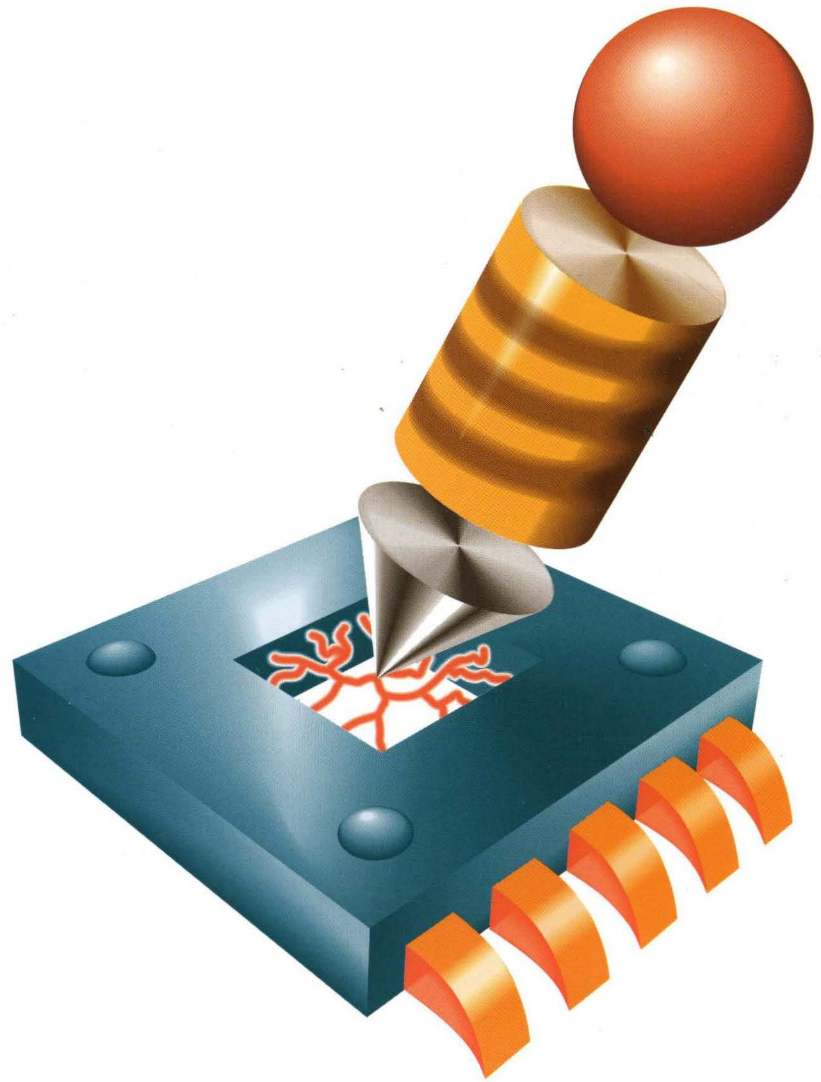
XILINX®

10/95

XILINX®

VIEWLOGIC

INTERFACE GUIDE



XACT™ STEP

October, 1995

Viewlogic Interface Guide

Viewlogic Interface Guide — 0401307 01

Printed in U.S.A.

Viewlogic Interface Guide

Preface

About This Manual

This manual explains how to use the Viewlogic interface software to translate your FPGA or EPLD designs from Viewlogic schematics to implemented design and simulation files.

Before using this manual, you should be familiar with the operations that are common to all of Xilinx's software tools: how to bring up the system, select a tool for use, specify operations, and manage design data. These topics are covered in the *Development System Reference Guide*.

Other publications that you can consult for related information are the *Viewlogic Tutorials* manual, the *Xilinx ABEL User Guide*, the *X-BLOX Reference/User Guide*, the *Timing Analyzer Reference/User Guide*, the *Design Manager/Flow Engine Reference/User Guide*, and the *XEPLD Reference Guide*. You can also refer to the PRO Series documentation from Viewlogic.

Manual Contents

This manual covers the following topics.

- Chapter 1, "Introduction," describes the programs that compose the Viewlogic interface, the Xilinx devices that Viewlogic's flow manager supports, and Viewlogic's design flow for FPGAs and EPLDs.
- Chapter 2, "Getting Started," explains how to configure your system to use PRO Series on PCs. It also briefly discusses how to use the Xilinx interface with Powerview on workstations.

- Chapter 3, "Design Entry," describes how to use PROcapture to enter a design.
- Chapter 4, "Functional Simulation," describes how to use PROflow to create a simulation network and how to use PROsim to perform a functional simulation.
- Chapter 5, "Design Implementation," describes how to use the Xilinx Design Manager to implement and create timing simulation data for FPGA and EPLD designs.
- Chapter 6, "Timing Simulation," describes how to use PROflow to create a simulation network and how to use PROsim to perform a timing simulation.
- Chapter 7, "Design and Simulation Techniques," discusses aspects of schematic entry and simulation with PRO Series and Powerview that you should be familiar with to use these tools effectively.
- Chapter 8, "Manual Translation," describes how to execute the XSimMake program, which automatically simulates your design; XMake, which automatically implements an FPGA design; and XEMake, which automatically implements an EPLD design. It also describes how to invoke the individual translation programs that produce a VSM file for simulation when you want to create it manually rather than automatically with XSimMake. In addition, this chapter explains how to use ViewGen to generate a ViewDraw schematic from a single WIR file.
- Chapter 9, "PROcapture Commands," lists and describes all the commands available in PROcapture.
- Chapter 10, "PROsim Commands," lists and describes all the commands available in PROsim.
- Chapter 11, "PROwave Commands," lists and describes all the commands available in PROwave.
- Appendix A, "Glossary," defines the key terms and concepts used throughout this manual.
- Appendix B, "Program Options," describes the options available in the translators that Viewlogic invokes to process your design manually.

- Appendix C, "Warning and Error Messages," lists the warning and error messages that you may receive from the translators used to process your design manually.
- Appendix D, "Altran," gives instructions for using Altran, the Library Alias Maintenance Facility, to change the library aliases when you port a design from one family to another while using the Unified Libraries.

Conventions

The following conventions are used in this manual's syntactical statements.

`Courier font regular`	System messages or program files appear in regular Courier font.
`Courier font bold`	Literal commands that you must enter in syntax statements are in bold Courier font.
italic font	Variables that you replace in syntax statements are in italic font.
`[ ]`	Square brackets denote optional items or parameters. However, in bus specifications, such as bus [7:0], they are required.
`{ }`	Braces enclose a list of items from which you must choose one or more.
⋮	A vertical ellipsis indicates material that has been omitted.
`...`	A horizontal ellipsis indicates that the preceding can be repeated one or more times.
|	A vertical bar separates items in a list of choices.
	This symbol denotes a carriage return.

Contents

Viewlogic Interface Guide

Introduction

Printed in U.S.A.

Chapter 1

Introduction

This chapter describes the programs that comprise the Viewlogic interface, the Xilinx devices that Viewlogic's flow manager supports, and Viewlogic's design flow for FPGAs and EPLDs.

Viewlogic Interface Programs

The Viewlogic interface programs support all of the Viewlogic systems: PRO Series, Powerview, and Workview PLUS. However, only PRO Series and some aspects of Powerview are discussed in this manual.

PRO Series

The PRO Series software is a windows-based system with a graphical user interface. It is available on PCs only.

This release supports PRO Series version 6.*x* and later.

Workview PLUS

Workview PLUS is windows-based Viewlogic software with a graphical user interface similar to Powerview's. It is available on PCs only.

Powerview

For workstations, Viewlogic offers Powerview, which contains enhancements in the graphical user interface.

PROcapture

PROcapture is Viewlogic's schematic entry tool for PRO Series.

PROsim

PROsim is Viewlogic's interactive logic simulator.

PROwave

PROwave is a waveform viewer and editor.

PROgen

PROgen is a utility that generates a PROcapture schematic from a WIR file.

PROsynthesis

PROsynthesis is Viewlogic's VHDL synthesis tool for the PC. It includes the VHDL designer and analyzer for synthesizing and simulating VHDL. PROsynthesis supports behavioral, RTL, and structural VHDL design styles and a number of design entry methods.

You can use PROsynthesis with XC2000, XC3000, XC3100, XC4000, XC5200, and XC7000 FPGAs.

The PROsynthesis software is available in the Viewlogic Stand-alone Extended packages from Xilinx or Viewlogic.

Xilinx Device Support

The Viewlogic flow manager supports XC2000/L, XC3000/A/L, XC3100/A, XC4000/A/H/D, XC5200, XC7200, and XC7300 devices. XC7200 and XC7300 are EPLD devices, and the other families are FPGA devices.

Design Flow

Creating FPGA and EPLD designs with Viewlogic PRO Series involves the following steps.

1. Enter your design with the PROcapture schematic editor, making sure that you observe the Xilinx design requirements noted in this manual.
2. Test the functionality of your design by creating a functional simulation network (VSM file) and loading it into PROsim to simulate the design. You can use PROwave to view the waveforms generated by the simulation.
3. Implement your FPGA or EPLD design using the Xilinx Design Manager.
4. Verify the timing of your design by creating a timing simulation network (VSM file) and loading it into PROsim to simulate the design. You can use PROwave to view the waveforms generated by the simulation.
5. Download the design and verify the board.

Figure 1-1 illustrates the design methodology for both FPGAs and EPLDs.

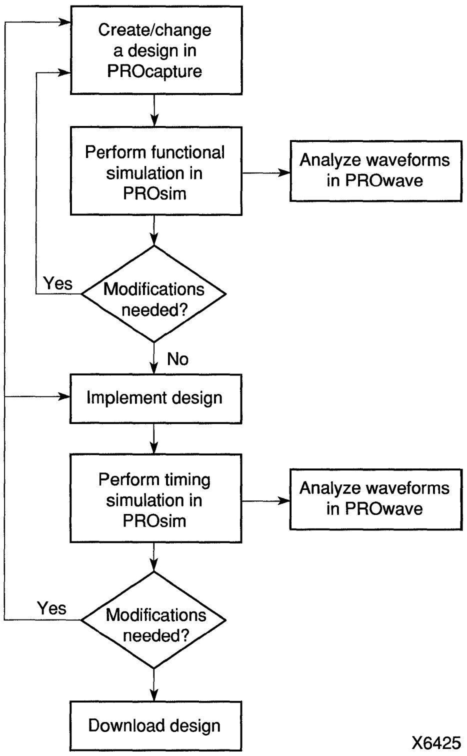

Create/change a design in PROcapture
Perform functional simulation in PROsim
Analyze waveforms in PROwave
Yes
Modifications needed?
No
Implement design
Perform timing simulation in PROsim
Analyze waveforms in PROwave
Yes
Modifications needed?
Download design
X6425

Figure 1-1 Viewlogic Design Methodology

Tutorials

The *Viewlogic Tutorials* manual contains tutorials that illustrate how to use PROcapture, PROsim, and PROwave for design entry and simulation. It also contains tutorials describing how to use PRO Series with Xilinx ABEL, X-BLOX, XACT-Performance, and the Timing Analyzer.

Viewlogic Interface Guide

Getting Started

Chapter 2

Getting Started

This chapter discusses how to configure your system to use PRO Series on PCs. It also briefly discusses how to use the Xilinx interface with Powerview on workstations.

PRO Series

For examples showing how to set up your PRO Series software, see the *Viewlogic Tutorials* manual.

Note: The instructions in this manual are written for the PRO Series user, and the PRO Series 6.*x* environment is shown in the figures of this document. However, you can use the Xilinx interface programs with any current Viewlogic software, including Powerview V5.3.*x* or Workview PLUS V6.*x*. Workview 4.1.3a is also supported but does not run under the Windows environment.

Required Software

To run PRO Series, you will need the following versions of the development software:

- PRO Series release 6.0 or later
- Xilinx/Viewlogic Interface and Libraries: WIR2XNF V6.0.*x* and XNF2WIR V6.0.*x* or later
- XACT*step* Development System Software: DS-502 V6.0.*x* or later for FPGAs on PCs; DS502 V5.2.*x* for FPGAs on workstations; DS-550 V6.0.*x* or later for EPLDs on PCs; DS-550 V5.2.*x* or later for EPLDs on workstations.

Installing the Software

Before you can use the PRO Series software, you must set up your PC to use the Viewlogic and XACT*step* Development System software.

1. Verify that your system is properly configured. Consult the Xilinx *Getting Started & Installation Guide* for instructions on setting up your machine to run the software.
2. Install one of the following sets of software. Each of these options includes Viewlogic PRO Series, the Xilinx/Viewlogic Interface and Libraries, and an XACT*step* Development System.
   - Base (DS-VLS-BAS-PC1), Standard (DS-VLS-STD-PC1), or Extended (DS-VLS-EXT-PC1) Stand-Alone (/S) Package Solutions for Viewlogic

     or
   - Viewlogic PROcapture Schematic Editor, Interface, and Libraries (DS-390), PROsim Simulator (DS-290), and XACT*step* Development System (DS-502) for FPGAs, and/or XACT*step* Development System (DS-550) for EPLDs

     or
   - Viewlogic PRO Series V6.*x* or later

     and

     Viewlogic Interface and Libraries (DS-391), and XACT*step* Development System (DS-502) for FPGAs and/or XACT*step* Development System (DS-550) for EPLDs; or Base (DS-VL-BAS-PC1) or Standard (DS-VL-STD-PC1) Interface Package Solution for Viewlogic
3. Verify that the following variables are set in your autoexec.bat file. It is assumed that you have loaded the software noted in the previous step to the c:\proser and c:\xact directories on your PC. If the software has been installed in different areas, modify the following Set statements accordingly. See the Xilinx *Getting Started & Installation Guide* for additional information on system setup.
   - The PATH variable sets the overall executable search path. It must include the directories where the PRO Series and XACT*step* Development System software have been installed. Use the following syntax.

PATH=*other_paths***;c:\XACT;c:\PROSER;***other_paths*

Note: The PATH variable cannot include any previous version of either the XACT*step* or Viewlogic software. Be sure to remove all paths to older software.

- The XACT variable is used by the XACT*step* and PRO Series software to locate data files. It must include the directory where the XACT*step* Development System resides and the directory that contains the \unified directory, where the Unified Libraries reside. Use this syntax:

  ```
  SET XACT=C:\XACT;C:\PROSER
  ```

Note: As with the PATH variable, you can set XACT to multiple directories using a semicolon (;) between the paths. In the syntax just given, the XACT*step* software is located in c:\xact, and the \unified directory is located in c:\proser. Because both paths are needed, they have been concatenated into a single path using a semicolon.

- The WDIR variable sets the data file search path for the PRO Series software. It must include a directory in which you can write. Use this syntax:

  ```
  SET WDIR=C:\PROSER\STANDARD
  ```

- The SYSPLT variable sets the PRO Series plotting directory. It must include a directory in which you can write. Use this syntax:

  ```
  SET SYSPLT=C:\PROSER\STANDARD
  ```

Starting Xilinx PROflow

Xilinx PROflow is a flow manager that guides you through all the steps involved in processing a design with PRO Series. On the basis of information such as the design type, part type, family, and the components used, PROflow determines which options are available and which programs it must run to process the design correctly. Every tool and process step is executed by or invoked from PROflow. In addition to managing the processing of your design, PROflow also seamlessly integrates all the tools needed to enter, implement, simulate, and download your design.

PROflow also handles design maintenance. When you initially select a design within PROflow, it creates a design project and defines the associated libraries.

1. To open Xilinx PROflow, double-click on the Xilinx PROflow icon, shown in Figure 2-1, in the Program Manager XACT*step* program group.

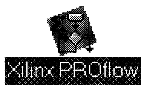
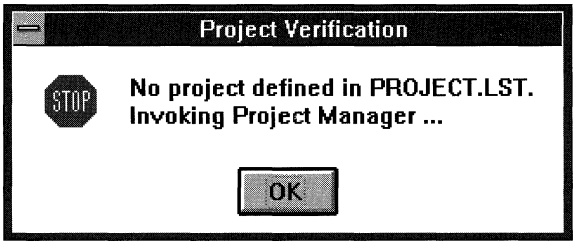


Figure 2-1 Xilinx PROflow Icon

Selecting the Xilinx PROflow icon for the first time may bring up a warning message, shown in Figure 2-2, stating that a project is not defined.



No project defined in PROJECT.LST.
Invoking Project Manager ...

Figure 2-2 Project Verification Warning Message

2. Click on **OK**.

   The message box closes, and the PRO Series Project Manager appears, as shown in Figure 2-3. The next describes how to use the Project Manager to create a project.

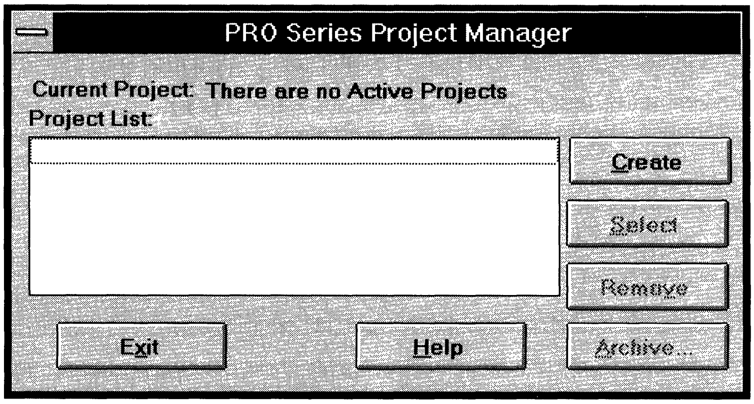

PRO Series Project Manager
Current Project: There are no Active Projects
Project List:
Create
Select
Remove
Exit
Help
Archive...

Figure 2-3 PRO Series Project Manager

Creating a Project

The Viewlogic tools use the concept of projects to keep track of designs. A project is a working directory that contains the sub-directories and data files for a given design. Projects can even contain several designs of the same type, for example, a single project containing several XC3000A designs; however, it is recommended that each project contain only one design. The project containing the design actively being processed is known as the current project.

The PRO Series Project Manager allows you to select, create, and remove projects.

1. Click on `Create` in the PRO Series Project Manager.

   The Create Project dialog box appears displaying the drive on which the software was loaded in the Directory field; Figure 2-4 shows an example.

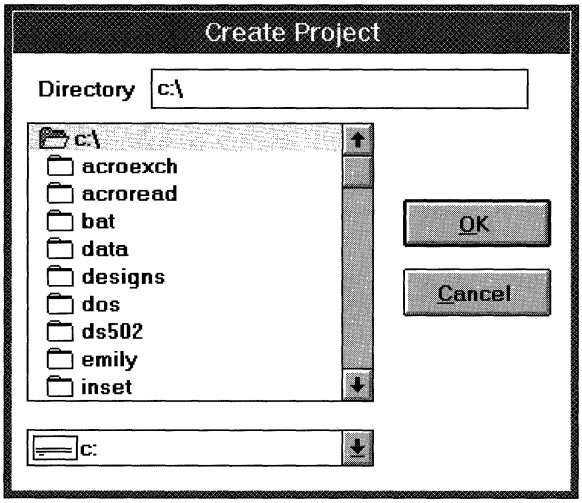


Figure 2-4 Create Project Dialog Box

2. Select the directory by double-clicking in the Directory list box until the Directory field is correctly updated, as illustrated in Figure 2-5.

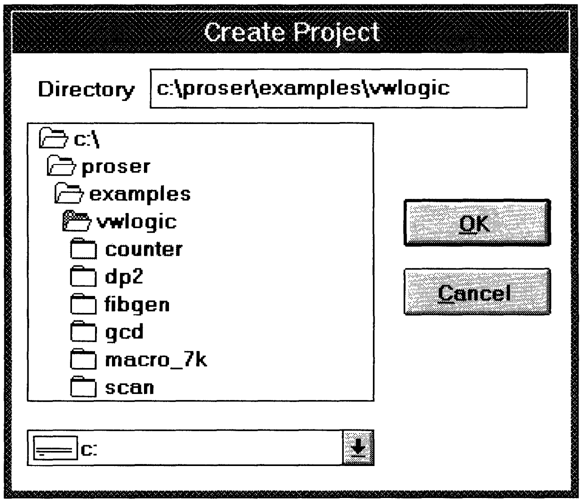


Figure 2-5 Updated Create Project Dialog Box

3. Click on **OK**.

   The Create Project dialog box closes, and the PRO Series Project Manager opens showing the selected directory in the Project List box, as indicated in Figure 2-6.

Note: If you are using a pre-existing project, the Project Manager displays a dialog box asking if the current project setup should be kept. In most cases, the current settings in the viewdraw.ini file should be kept.

For new projects, the Project Manager automatically creates the necessary initialization files when you create the new project.

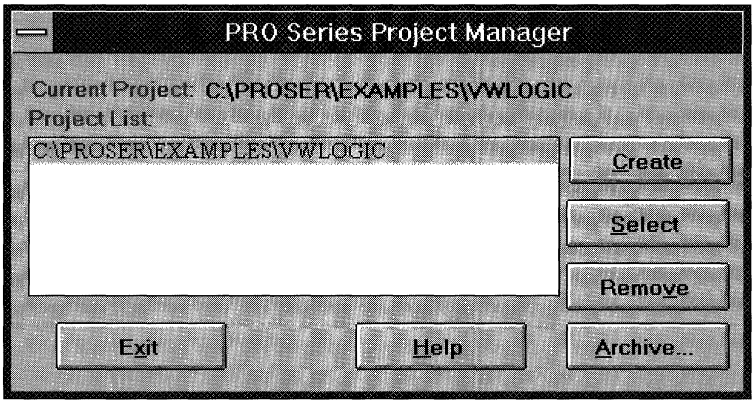
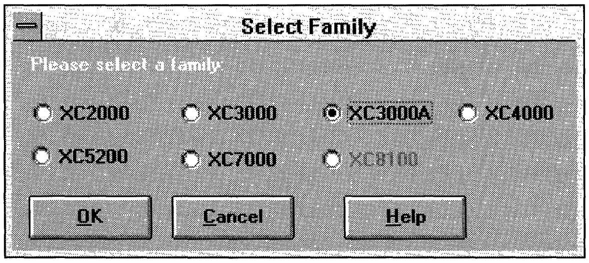

PRO Series Project Manager
Current Project: C:\PROSER\EXAMPLES\VWLOGIC
Project List
C:\PROSER\EXAMPLES\VWLOGIC
Create
Select
Remove
Exit
Help
Archive...

Figure 2-6 Updated PRO Series Project Manager

4. Click on **`Select`**.
5. To close the PRO Series Project Manager, click on **`Exit`**.

   When you close the PRO Series Project Manager, the Select Family dialog box appears, as shown in Figure 2-7.



Please select a family: XC2000 XC3000 [✓ XC3000A] XC4000 XC5200 XC7000 XC8100

Figure 2-7 Select Family Dialog Box

6. Select the desired family for the project.
7. Click on **OK**.

   The Xilinx PROflow window appears, as shown in Figure 2-8.

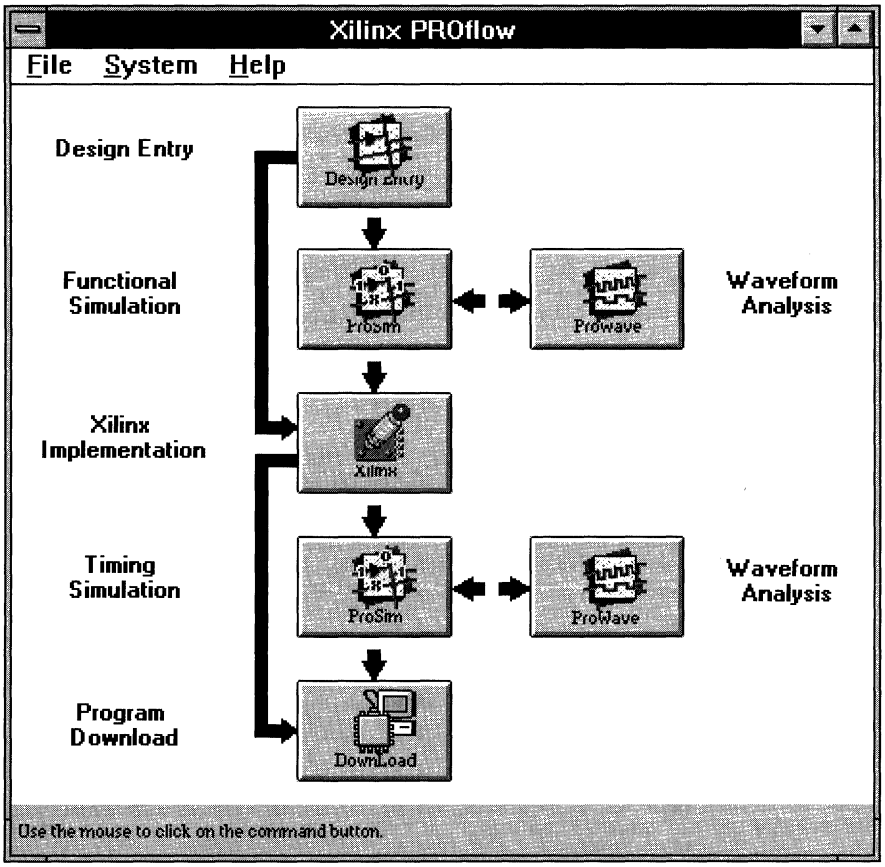
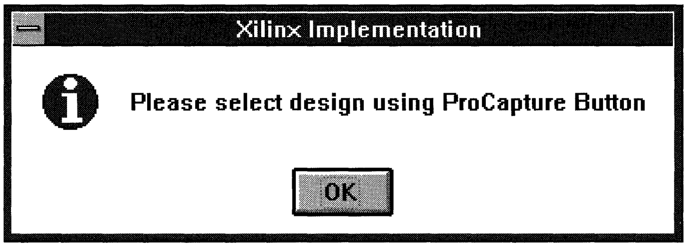


Figure 2-8 Xilinx PROflow Window

To guide you through the processing, PROflow only allows you to enter stages if the files that are needed for that step are present. For instance, if you click on the Xilinx Implementation icon, the message shown in Figure 2-9 appears.



Please select design using ProCapture Button

Figure 2-9 Xilinx Implementation Warning Message

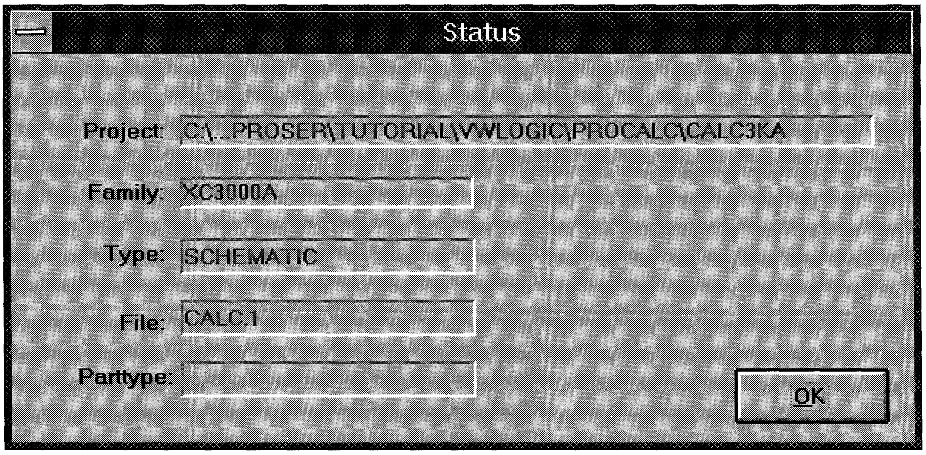
Because the design to be processed has not been defined, PROflow does not allow you to invoke the Xilinx Implementation section.

Obtaining Design Status

To see PROflow's initial project status, follow these steps.

1. Select the **`File`** → **`Status`** command.

   This command displays the Status dialog box, shown in Figure 2-10, which displays the currently known information about the design. As you enter and process the design, additional information appears in the remaining fields.



Project: C:\...PROSER\TUTORIAL\VWLOGIC\PROCALC\CALC3KA
Family: XC3000A
Type: SCHEMATIC
File: CALC.1
Parttype:

Figure 2-10 Status Dialog Box

2. Click on **OK**.

   The Status dialog box closes.

Customizing Your Environment and Library Search Order

The viewdraw.ini file is the PROcapture initialization file. It contains the setup parameters for PROcapture as well as the libraries being used for the designs in the project. You must keep a copy of the viewdraw.ini file in each project directory so you can customize the library search order for each project. The viewdraw.ini file in \proser\standard is overwritten each time that you update the software, so any changes that you made to that file are lost.

PROflow creates a standard viewdraw.ini file automatically when a project is defined. The following sections describe how to change this standard setup to include user-created libraries and references to other projects.

Library Sequential Search Order

The library sequential search order format allows you to specify an unlimited number of directories, with only one primary directory, and the order in which you want the library directories searched. The search order is determined by the order in which the directories are listed in the viewdraw.ini file.

It is strongly recommended that you place the primary directory first in the search path order. The directories are categorized as follows:

p — Primary directory
w — Secondary read/write directories
r — Secondary read-only directories
m — Megafile read-only directories

Library directories should conform to the search order shown in Table 2-1.

Table 2-1 Library Search Order

Type	Directory	Library
p	\project_directory	
m	\proser\shm4000	shm4000
m	\proser\unified\xc2000	xc2000
m	\proser\unified\xc3000	xc3000
m	\proser\unified\xc4000	xc4000
m	\proser\unified\xc5200	xc5200
m	\proser\unified\xc7000	xc7000
m	\proser\unified\xblox	xblox
m	\proser\unified\builtin	builtin
m	\proser\unified\xbuiltin	xbuiltin

Note: The Xilinx Viewlogic library package includes the builtin library, which is a simplified version of the complete builtin library sold by Viewlogic. You should use only the Xilinx-supplied builtin library (\proser\unified\builtin) in any Xilinx project directory.

You must add the builtin and xbuiltin libraries to the viewdraw.ini file if you simulate, and they must be specified *last*. These libraries are the Viewlogic simulation model libraries. If you do not specify them, you cannot simulate or push into the primitives to view the simulation models. Neither the builtin nor the xbuiltin libraries can be used to generate a user-created primitive library.

Ensure that the libraries associated with only one family appear in your viewdraw.ini file when you are entering a design. See the "Modifying the Viewdraw.ini File" section later in this chapter for information on how to modify the libraries.

Note: Different family libraries cannot be mixed in the same project directory. Specify libraries from only one Xilinx family in the viewdraw.ini file for each project directory.

For an XC4000 design on a PC, the library definition section should look like the following example.

```
 DIR [p] .                               (primary)
 DIR [m] \proser\shm4000                 (shm4000)
|DIR [m] \proser\unified\xc2000          (xc2000)
|DIR [m] \proser\unified\xc3000          (xc3000)
 DIR [m] \proser\unified\xc4000          (xc4000)
|DIR [m] \proser\unified\xc5200          (xc5200)
|DIR [m] \proser\unified\xc7000          (xc7000)
 DIR [m] \proser\unified\xblox           (xblox)
 DIR [m] \proser\unified\builtin         (builtin)
 DIR [m] \proser\unified\xbuiltin        (xbuiltin)
|DIR [m] \proser\unified\xc2000o         (xc2000o)
|DIR [m] \proser\unified\xc3000o         (xc3000o)
 DIR [m] \proser\unified\xc4000o         (xc4000o)
|DIR [m] \proser\unified\xc5200o         (xc5200o)
|DIR [m] \proser\unified\xc7000o         (xc7000o)
```

Note: Specify the xc*xx*00o libraries only if you would like to know which components in the Unified Libraries replaced the components in the libraries pre-dating the Unified Libraries. The xc*xx*00o libraries are merely "notes" describing the current component names.

Specifying Library Aliases

When specifying the library search order, you must also add a library alias to each library directory. A library alias is a name that identifies a specific library directory along with the parts that it contains. The alias distinguishes identically named components from different libraries. You must specify in the viewdraw.ini file the aliases for each Xilinx library for proper netlist translation.

Using library aliases, you can distinguish symbols with the same name from different libraries on a single schematic. Since only components from Xilinx libraries can be placed in a Xilinx schematic, you use this feature only when you have a custom library or when you are creating a system-level schematic containing different kinds of chips.

The library alias must be specified in parentheses in the viewdraw.ini file. Do not substitute any other aliases, because these alias names are used in the macro schematics within each library. XC3000, XC3000A/L, and XC3100A families use the XC3000 alias; XC4000 and XC4000A/D/H families use the XC4000 alias; XC5200 families use the XC5200 alias, and XC7200 and XC7300 families use the

XC7000 alias. Following are the libraries to use with each of the Xilinx families; notice the aliases specified for each library.

- XC2000

```
DIR [m]  \proser\unified\xc2000 (xc2000)
DIR [m]  \proser\unified\builtin (builtin)
DIR [m]  \proser\unified\xbuiltin (xbuiltin)
DIR [m]  \proser\unified\xc2000o (xc2000o)
```

- XC3000

```
DIR [m]  \proser\unified\xc3000 (xc3000)
DIR [m]  \proser\unified\xblox (xblox)
DIR [m]  \proser\unified\builtin (builtin)
DIR [m]  \proser\unified\xbuiltin (xbuiltin)
DIR [m]  \proser\unified\xc3000o (xc3000o)
```

- XC4000

```
DIR [m]  \proser\shm4000 (shm4000)
DIR [m]  \proser\unified\xc4000 (xc4000)
DIR [m]  \proser\unified\xblox (xblox)
DIR [m]  \proser\unified\builtin (builtin)
DIR [m]  \proser\unified\xbuiltin (xbuiltin)
DIR [m]  \proser\xc4000o (xc4000o)
```

- XC5200

```
DIR [m]  \proser\unified\xc5200 (xc5200)
DIR [m]  \proser\unified\xblox (xblox)
DIR [m]  \proser\unified\builtin (builtin)
DIR [m]  \proser\unified\xbuiltin (xbuiltin)
DIR [m]  \proser\xc5200o (xc5200o)
```

- XC7000

```
DIR [m]  \proser\unified\xc7000 (xc7000)
DIR [m]  \proser\unified\builtin (builtin)
DIR [m]  \proser\unified\xbuiltin (xbuiltin)
DIR [m]  \proser\xc7000o (xc7000o)
```

On a PC, you can add drive designators to the library paths, if necessary. Otherwise, the drive on which the viewdraw.ini file resides is assumed.

Modifying the Viewdraw.ini File

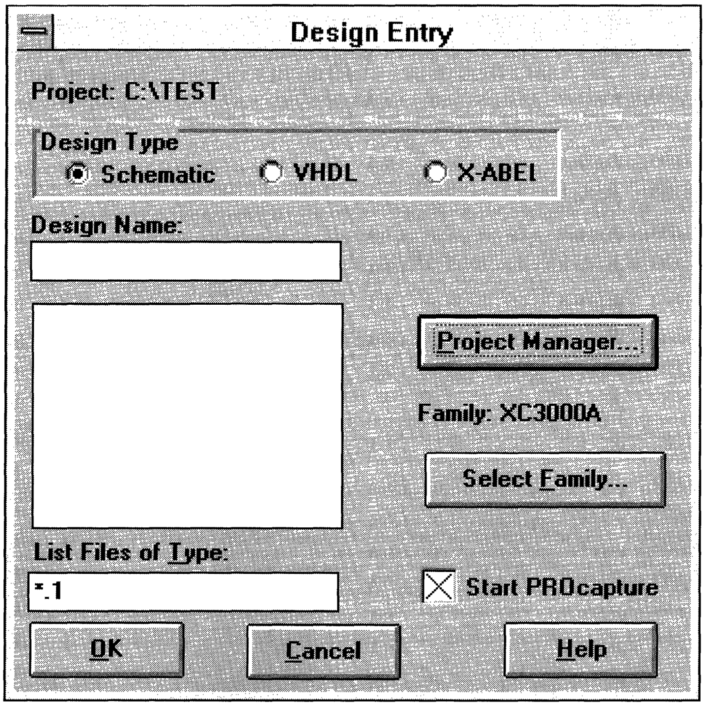
When you initially create your project, PROflow creates the viewdraw.ini file for you. However, you may want to customize your viewdraw.ini file in certain cases, for example, when you have created a library containing macros that you want to use on a schematic. The Library List Editor is a utility that makes updating the viewdraw.ini file easier.

Invoking the Library List Editor

To invoke the Library List Editor, follow these instructions.

1. In PROflow, click on the Design Entry icon.

   The Design Entry dialog box appears, as shown in Figure 2-11.


Design Entry

Project: C:\TEST

Design Type: [✓ Schematic] VHDL X-ABEL

Design Name:

List Files of Type: *.1

Family: XC3000A

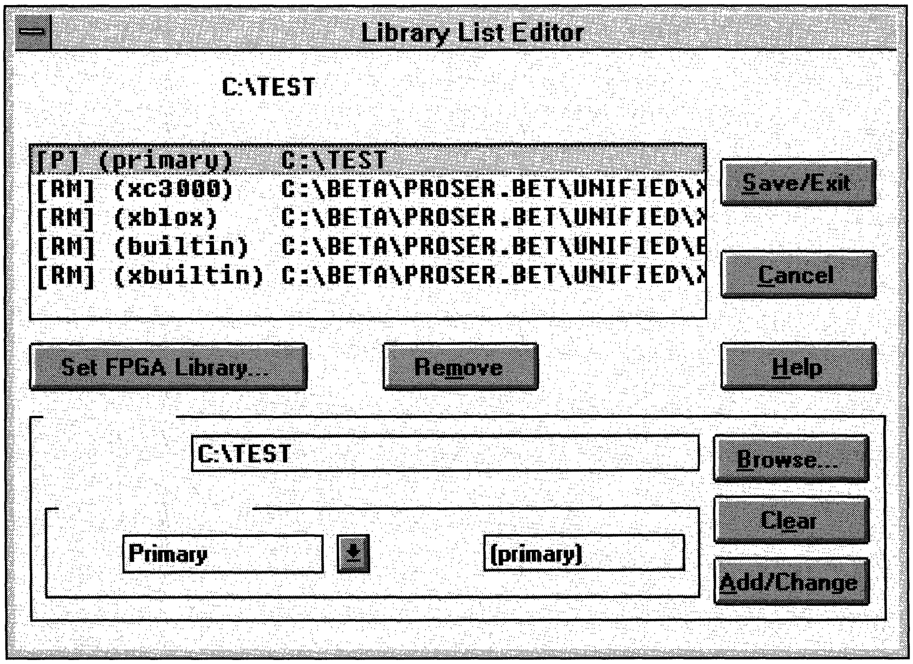
[✓ Start PROcapture]

Figure 2-11 Design Entry Dialog Box

2. Enter the name of the design in the Design Name field.
3. Click on **OK**.

   The PROcapture window now appears.
4. Select the **File** → **Close** command to close the current schematic.
5. In PROcapture, click on **File** → **Library List Editor**.

   The Library List Editor appears, as illustrated by the example in Figure 2-12. In the Current Libraries list box, the first column displays the format, the second shows the aliases, and the third lists the library path names. The following section explains the syntax of the viewdraw.ini file.



```
[P] (primary)   C:\TEST
[RM] (xc3000)   C:\BETA\PROSER.BET\UNIFIED\X
[RM] (xblox)    C:\BETA\PROSER.BET\UNIFIED\X
[RM] (builtin)  C:\BETA\PROSER.BET\UNIFIED\E
[RM] (xbuiltin) C:\BETA\PROSER.BET\UNIFIED\X
```

Figure 2-12 Library List Editor

Viewdraw.ini File Syntax

Near the end of the viewdraw.ini file is a series of lines defining the path names for each component library directory. These lines have the following syntax:

DIR [*format*] *path* (*alias*)

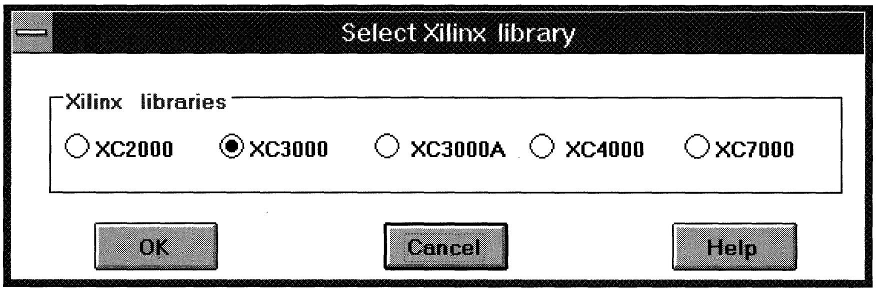
- *Format* indicates the library directory format; it can be one of the following.
  - [m] indicates that the library is compressed into a megafile format. Megafiles are read-only by default. The Xilinx-supplied libraries are in megafile format for the PC.
  - [r] indicates a read-only directory.
  - [w] indicates a read-write directory.
  - [p] denotes the primary or project directory. All symbols and schematics that you create are saved here.
- *Path* is the full path specification of the library. Specifying a period (.) as the path name for the primary ([p]) directory causes PRO Series to use the project directory as the base directory for all new schematics, user-generated symbols, netlists, and related simulation files.
- *Alias* is the library name associated with each component that you place in your schematic.

Adding Xilinx Libraries

Now you are ready to modify the viewdraw.ini file.

1. To add the standard Xilinx libraries to your project, click on **Set FPGA Library**.

   The Select Xilinx Library dialog box appears, as shown in Figure 2-13.



Selected: [● XC3000]

Figure 2-13 Select Xilinx Library Dialog Box

2. Click on the desired FPGA family.
3. Click on **OK**.

Adding and Removing Libraries

You can optionally add your own libraries to the viewdraw.ini file.

1. To add an individual library, highlight the line above the location of the library to be added, and use either of the following methods.
   - Type the library path name in the **`Library Path`** field, the format in the **`Type`** field, and the library alias in the **`Alias`** field. You must include the parentheses when typing in the alias.
   - Alternatively, you can click on **`Browse`**, select the desired library, and click on **OK**. The Type and Alias fields are automatically updated.
2. Click on **`Add/Change`**.
3. Use the **`Remove`** button to delete any libraries that are out of order, and repeat steps 1 and 2 to add them in the correct order.
4. Click on **`Save/Exit`** to save your changes to the viewdraw.ini file and exit the Library List Editor dialog box.

Note: You can use the Clear button to clear the three fields in the Edit Library panel.

Powerview

This section describes how to set up your Powerview environment.

Required Software

To run Powerview, you will need the following versions of the development software:

- Viewlogic Interface and Libraries, DS-391: V6.0.*x* or later.

Installing the Software

Before you can use the Powerview software, you must set up your workstation to use the Viewlogic and XACT*step* Development System software.

1. Verify that your system is properly configured. Consult the Xilinx *Getting Started & Installation Guide* for instructions on setting up your machine to run the software.
2. Install the DS-391 package or the DS-VL Standard package.

Creating a Project

You can create a Viewlogic project directory anywhere on your system, but you must follow a standard structure within that directory. Each project directory must contain the sch, sym, and wir subdirectories.

- The sch directory contains files with graphic descriptions of your schematic.
- The sym directory contains files with graphic descriptions of the symbols that you create and save.
- The wir directory contains files with netlist descriptions of schematics.

The Viewlogic project directory structure is shown in Figure 2-14.

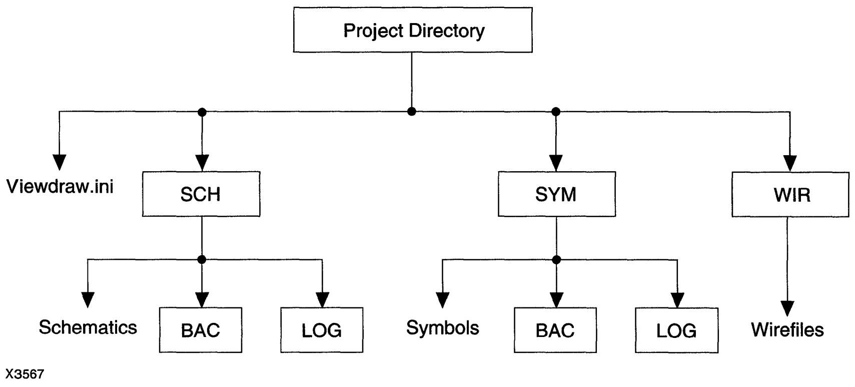


Figure 2-14 PROcapture Directory Structure

Before creating a design, you can optionally use ViewFile, a project management utility, to create a project directory in which the design files will reside.

You can create several project directories. If you are using a machine on which someone else has already used ViewFile, there will already be other project directories.

Customizing Your Environment and Library Search Order

Like PRO Series, Powerview uses the viewdraw.ini file to set up the ViewDraw environment. This file also contains the path and directory search order for your libraries. You must keep a copy of the viewdraw.ini file in each project directory, so you can customize the library search order for each project. The viewdraw.ini file in /*powerview_path*/standard is overwritten each time that you update the software, so any changes that you made to that file are lost.

Library Sequential Search Order

The library sequential search order format allows you to specify an unlimited number of directories, with only one primary directory, and the order that you want the library directories to be searched. The search order is specified by the order in which the directories are listed in the viewdraw.ini file.

It is strongly recommended that you place the primary directory first in the search path order. The directories are categorized as follows:

p — Primary directory
w — Secondary read/write directories
r — Secondary read-only directories
m — Megafile read-only directories

Library directories should conform to the search order shown in Table 2-2.

Table 2-2 Library Search Order

Type	Directory	Library
p	/*project_directory*	
r	/*powerview_path*/shm4000	shm4000
r	/*powerview_path*/unified/xc2000	xc2000
r	/*powerview_path*/unified/xc3000	xc3000
r	/*powerview_path*/unified/xc4000	xc4000
r	/*powerview_path*/unified/xc5200	xc5200
r	/*powerview_path*/unified/XC7000	xc7000
r	/*powerview_path*/unified/xblox	xblox
r	/*powerview_path*/unified/builtin	builtin
r	/*powerview_path*/unified/xbuiltin	xbuiltin

Note: The Xilinx Viewlogic library package includes the builtin library, which is a simplified version of the complete builtin library sold by Viewlogic. You should use only the Xilinx-supplied builtin library (/*powerview_path*/unified/builtin) in any Xilinx project directory.

You must add the builtin and xbuiltin libraries to the viewdraw.ini file if you simulate, and they must be specified *last*. These libraries are the Viewlogic simulation model libraries. If you do not specify them, you cannot simulate or push into the primitives to view the simulation models. Neither the builtin nor the xbuiltin libraries can be used to generate a user-created primitive library.

Note: If you use the ViewDraw ViewFile system for project directory management, you must change your current Powerview project directory in a ViewFile window when you change your XDM directory. Otherwise, ViewDraw automatically changes to the last specified ViewFile project directory.

Do not forget to include the library alias names under the Library column. These aliases must be entered exactly as shown.

For an XC4000 design on a workstation, the library definition section should look like the next example:

```
 DIR [p] .                                 (primary)
 DIR [r] /powerview_path/shm4000           (shm4000)
|DIR [r] /powerview_path/unified/xc2000    (xc2000)
|DIR [r] /powerview_path/unified/xc3000    (xc3000)
 DIR [r] /powerview_path/unified/xc4000    (xc4000)
|DIR [r] /powerview_path/unified/xc5200    (xc5200)
|DIR [r] /powerview_path/unified/xc7000    (xc7000)
 DIR [r] /powerview_path/unified/xblox     (xblox)
 DIR [r] /powerview_path/unified/builtin   (builtin)
 DIR [r] /powerview_path/unified/xbuiltin(xbuiltin)
|DIR [r] /powerview_path/unified/xc2000o   (xc2000o)
|DIR [r] /powerview_path/unified/xc3000o   (xc3000o)
 DIR [r] /powerview_path/unified/xc4000o   (xc4000o)
|DIR [r] /powerview_path/unified/xc5200o   (xc5200o)
|DIR [m] /powerview_path/unified/xc7000o   (xc7000o)
```

Note: Specify the xc*xx*00o libraries only if you would like to know which components in the Unified Libraries replaced the components in the libraries pre-dating the Unified Libraries. The xc*xx*00o libraries are merely "notes" describing the current component names.

Specifying Library Aliases

When specifying the library search order, you must also add a library alias to each library directory. A library alias is a name that identifies a specific library directory along with the parts that it contains. The alias distinguishes identically named components from different libraries. You must specify in the viewdraw.ini file the aliases for each Xilinx library for proper netlist translation.

Using library aliases, you can distinguish symbols with the same name from different libraries on a single schematic. Since only components from Xilinx libraries can be placed in a Xilinx schematic, you use this feature only when you have a custom library or when you are creating a system-level schematic containing different kinds of chips.

The library alias must be specified in parentheses in the viewdraw.ini file. Do not substitute any other aliases, because these alias names are used in the macro schematics within each library. XC3000, XC3000A/L, and XC3100A families use the XC3000 alias; XC4000 and XC4000A/D/H families use the XC4000 alias; XC5200 families use the XC5200 alias, and XC7200 and XC7300 families use the XC7000 alias. Following are the libraries to use with each of the Xilinx families; notice the aliases specified for each library.

- XC2000

  DIR [r] /*powerview_path*/unified/xc2000 (xc2000)
  DIR [r] /*powerview_path*/unified/builtin (builtin)
  DIR [r] /*powerview_path*/unified/xbuiltin (xbuiltin)
  DIR [r] /*powerview_path*/unified/xc2000o (xc2000o)

- XC3000

  DIR [r] /*powerview_path*/unified/xc3000 (xc3000)
  DIR [r] /*powerview_path*/unified/xblox (xblox)
  DIR [r] /*powerview_path*/unified/builtin (builtin)
  DIR [r] /*powerview_path*/unified/xbuiltin (xbuiltin)
  DIR [r] /*powerview_path*/unified/xc3000o (xc3000o)

- XC4000

  DIR [r] /*powerview_path*/shm4000 (shm4000)
  DIR [r] /*powerview_path*/unified/xc4000 (xc4000)
  DIR [r] /*powerview_path*/unified/xblox (xblox)
  DIR [r] /*powerview_path*/unified/builtin (builtin)
  DIR [r] /*powerview_path*/unified/xbuiltin (xbuiltin)
  DIR [r] /*powerview_path*/xc4000o (xc4000o)

- XC5200

  DIR [r] /*powerview_path*/unified/xc5200 (xc5200)
  DIR [r] /*powerview_path*/unified/xblox (xblox)
  DIR [r] /*powerview_path*/unified/builtin (builtin)
  DIR [r] /*powerview_path*/unified/xbuiltin (xbuiltin)
  DIR [r] /*powerview_path*/xc5200o (xc5200o)

- XC7000

  DIR [r] /*powerview_path*/unified/xc7000 (xc7000)
  DIR [r] /*powerview_path*/unified/builtin (builtin)
  DIR [r] /*powerview_path*/unified/xbuiltin (xbuiltin)
  DIR [r] /*powerview_path*/xc7000o (xc7000o)

Viewdraw.ini File Syntax

Near the end of the viewdraw.ini file is a series of lines defining the path names for each component library directory. These lines have the following syntax:

DIR [*format*] *path* (*alias*)

- *Format* indicates the library directory format; it can be one of the following:
  - [m] indicates that the library is compressed into a megafile format. Megafiles are read-only by default. Workstation libraries are not compressed into this format.
  - [r] indicates a read-only directory. If you are a workstation user, specify this format for the Xilinx-supplied libraries.
  - [w] indicates a read-write directory.
  - [p] denotes the primary or project directory. All symbols and schematics that you create are saved here.
- *Path* is the full path specification of the library. Specifying a period (.) as the path name for the primary ([p]) directory causes ViewDraw to use the directory in which the file resides as the base directory for all new schematics, user-generated symbols, netlists, and related simulation files.
- *Alias* is the library name associated with each component that you place in your schematic.

Modifying the Viewdraw.ini File

You can use a text editor or Powerview's Cockpit utility to modify the viewdraw.ini file. Directions for modifying this file with a text editor are given in this section; refer to the Powerview documentation for instructions on using Cockpit.

To add a library with a text editor, place a line beginning with the DIR keyword in the viewdraw.ini file as follows:

`DIR [r]` */powerview_path*/**`unified`**/*library_name* (*alias*)

Note: Ensure that the libraries associated with only one family appear in your viewdraw.ini file when you are entering a design. Comment out the libraries for the other families by inserting a pipe character (|) at the beginning of the line.

Viewlogic Interface Guide

Design Entry

Chapter 3

Design Entry

This chapter describes how to use PROcapture to enter a design.

Invoking PROcapture

To start PROcapture, follow this procedure.

1. In the Xilinx PROflow window, click on the Design Entry icon, shown in Figure 3-1.

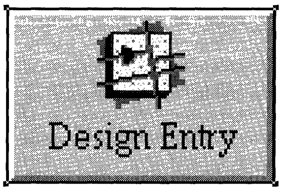
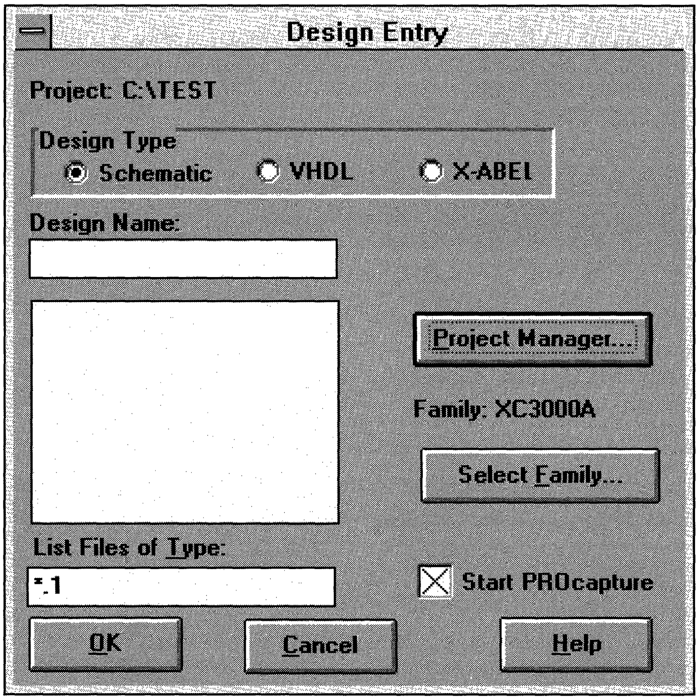


Figure 3-1 Design Entry Icon

The Design Entry dialog box now appears, as shown in Figure 3-2.


Design Entry

Project: C:\TEST

Design Type
[✓ Schematic] VHDL X-ABEL

Design Name:

Family: XC3000A

List Files of Type:
*.1

[✓ Start PROcapture]

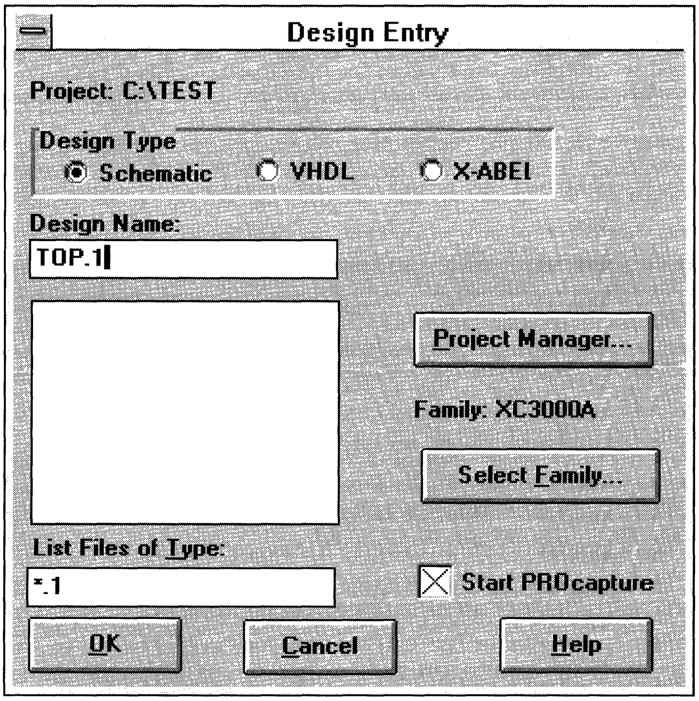
Figure 3-2 Design Entry Dialog Box

The project directory is displayed at the top of the dialog box. The List Files of Type field displays the default filter, *.1.

Viewlogic allows a schematic to contain multiple sheets. The sheets are saved to the project directory's sch directory, where each sheet is a separate file. The extension of the file is the actual sheet number. If the top-level schematic has two sheets, the sch directory would contain *schematic*.1 and *schematic*.2 files. The current filter in the List Files of Type field in the Design Entry dialog box restricts the display to only the first sheet of each schematic. If you wanted to see all the second sheets, you could change the List Files of Type field to *.2. To see all sheets, you could enter *.* in the List Files of Type field.

Note: The default of the Design Type field is Schematic. You can also process VHDL designs if the PROsynthesis package is installed, and ABEL designs if Xilinx ABEL is installed.

2. To select the top-level schematic, you can either type the name directly in the `Design Name` field, as shown in Figure 3-3, or click on the file in the design list box.



Project: C:\TEST

Design Type: [✓ Schematic] VHDL X-ABEL

Design Name: TOP.1

Family: XC3000A

List Files of Type: *.1

[✓ Start PROcapture]

Figure 3-3 Entering the Design Name

3. Verify that the Start PROcapture check box is selected.

   Clicking on the Start PROcapture check box determines whether or not PROcapture is invoked when you press the OK button. By default, Start PROcapture is toggled on.

4. Click on **OK**.

The PROcapture window now appears, as shown in Figure 3-4.

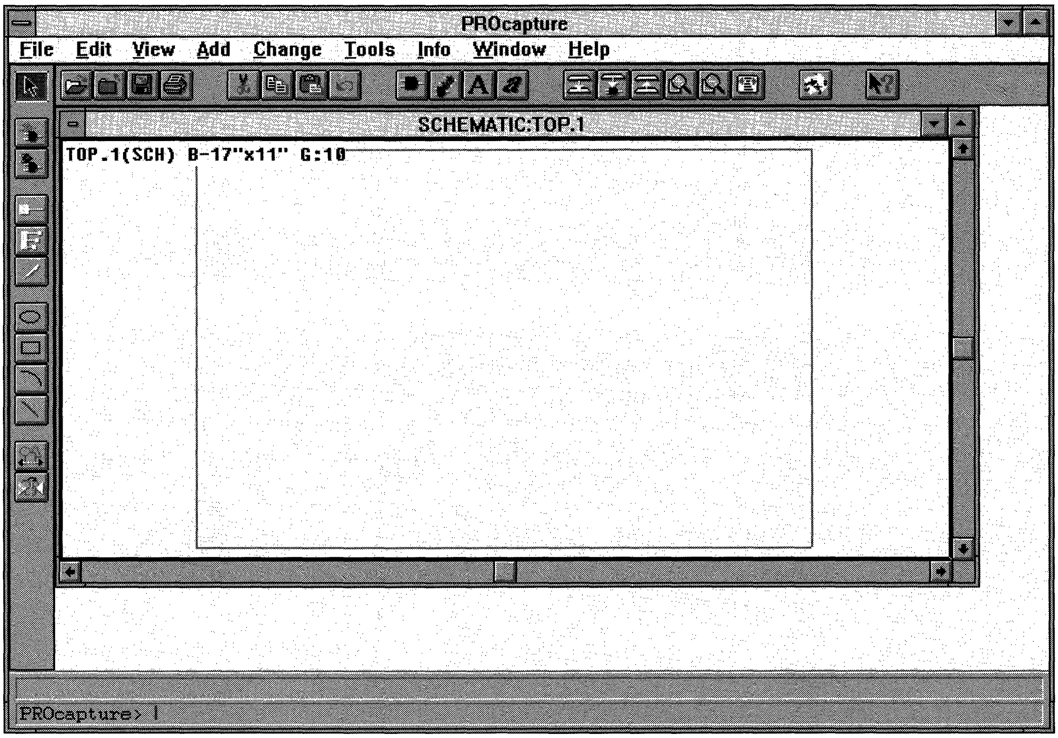

PROcapture
File Edit View Add Change Tools Info Window Help
SCHEMATIC:TOP.1
TOP.1(SCH) B-17"x11" G:10
PROcapture>

Figure 3-4 PROcapture Window

Exiting PROcapture

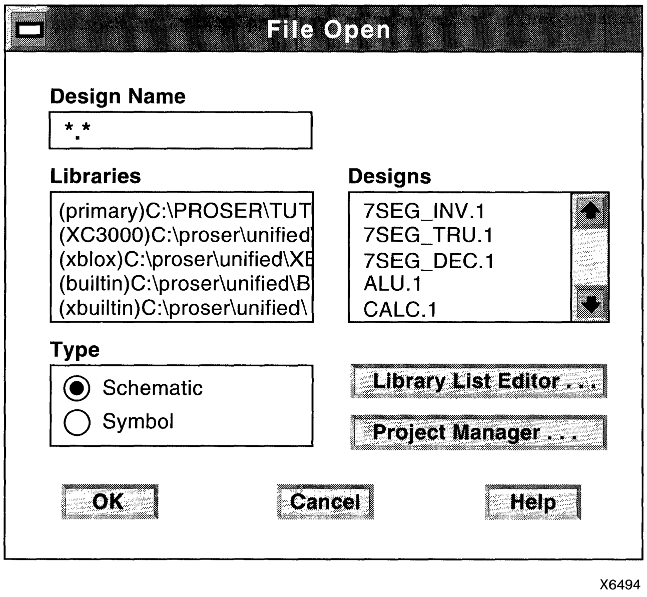
To exit PROcapture, select **`File`** → **`Exit`**.

Creating a Schematic

When you invoke PROcapture from PROflow, the initial schematic is opened for you. The following procedure demonstrates how to open and create subsequent schematics without having to return to PROflow.

1. Open a blank schematic window by clicking on **`File`** → **`Open`**.

   The File Open dialog box appears, as shown in Figure 3-5.



Design Name: `*.*`

Libraries:
```
(primary)C:\PROSER\TUT
(XC3000)C:\proser\unified
(xblox)C:\proser\unified\XE
(builtin)C:\proser\unified\B
(xbuiltin)C:\proser\unified\
```

Designs:
```
7SEG_INV.1
7SEG_TRU.1
7SEG_DEC.1
ALU.1
CALC.1
```

Type: [✓ Schematic] Symbol

Figure 3-5 File Open Dialog Box

Keyboard Shortcut: You can execute the File → Open command for a schematic by typing **`sch`**↵ on the PROcapture command line. You can also specify the schematic to be opened. For example, to open a schematic named ALU, type sch alu↵.

Toolbar Shortcut: You can execute the File → Open command by clicking on the Open toolbar icon, shown in Figure 3-6.

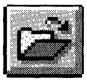


Figure 3-6 Open Toolbar Icon

2. In the **`Design Name`** field, type in the name of the new schematic.
3. Click on **`OK`**.
4. The blank schematic sheet now opens with the name of the new schematic at the top of the window, followed by a .1 extension.

Creating Schematics with Multiple Sheets

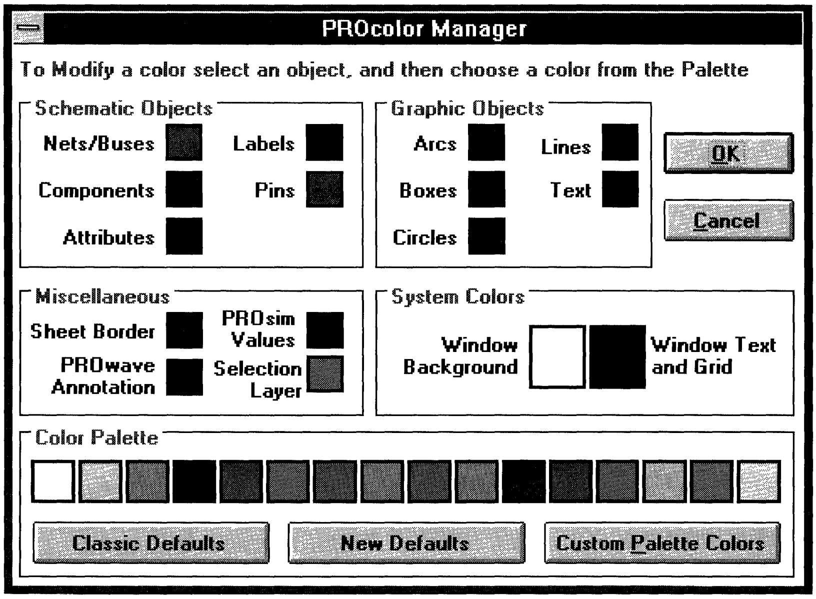
When you create multiple-sheet designs, each sheet must have the same file name with extensions .1, .2, .3, and so forth. This convention applies to both top-level and lower-level schematics. The default sheet number is 1; to open a different sheet, type the schematic name and sheet number separated by a period in the Design Name field, or click on the file in the Designs list box. To view each sheet, use the **`View`** → **`Push Next Sheet`** command from the menu or **`psh`**↵ from the keyboard.

Changing the PROcapture Window Colors

You can change the PROcapture color settings so that viewing a schematic will be easier. If the background of your schematic is white and you want to change the color palette, proceed with the steps in this section.

1. To change the color palette, select **`Change`** → **`PROcapture Colors`**.

   The PROcolor Manager dialog box appears, as shown in Figure 3-7.


PROcolor Manager

To Modify a color select an object, and then choose a color from the Palette

Figure 3-7 PROcolor Manager Dialog Box

2. Click on **`Classic Defaults`**, which renders PROcapture compatible with the Xilinx libraries.

   This setting changes the color configuration for the various objects. The window background and the window text and grid toggle from white to black and vice versa.

3. Click on **`OK`** to close the PROcolor Manager dialog box and reactivate PROcapture.

   An information dialog box comes up to inform you that the changes to the color palette will not take effect until you close all schematics and symbols.

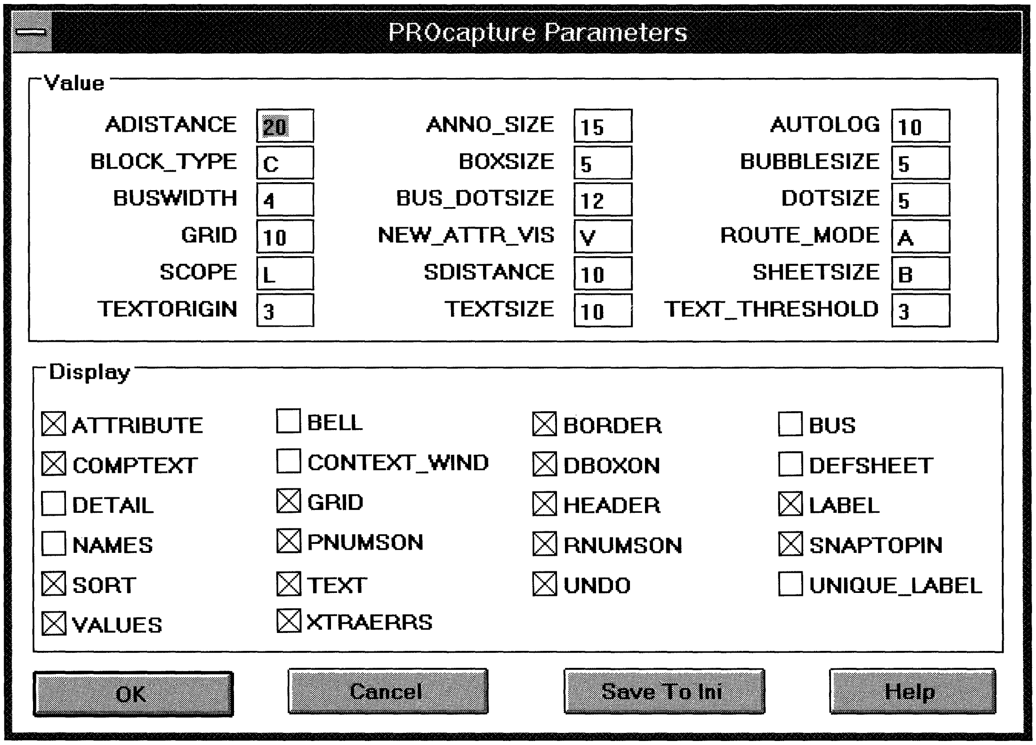
4. Select **`File`** → **`Close`** to close any open schematics.
5. Select **`File`** → **`Open`** to re-open the schematics.

Changing PROcapture Settings

PROcapture offers a dialog box that allows you to change various aspects of the PROcapture display and save these settings to the project's viewdraw.ini file.

1. Click on **`Change`** → **`PROcapture Parameters`**.

   The PROcapture Parameters dialog box appears, as indicated in Figure 3-8.



PROcapture Parameters

Value

ADISTANCE	20	ANNO_SIZE	15	AUTOLOG	10
BLOCK_TYPE	C	BOXSIZE	5	BUBBLESIZE	5
BUSWIDTH	4	BUS_DOTSIZE	12	DOTSIZE	5
GRID	10	NEW_ATTR_VIS	V	ROUTE_MODE	A
SCOPE	L	SDISTANCE	10	SHEETSIZE	B
TEXTORIGIN	3	TEXTSIZE	10	TEXT_THRESHOLD	3

Display

☒ ATTRIBUTE	☐ BELL	☒ BORDER	☐ BUS
☒ COMPTEXT	☐ CONTEXT_WIND	☒ DBOXON	☐ DEFSHEET
☐ DETAIL	☒ GRID	☒ HEADER	☒ LABEL
☐ NAMES	☒ PNUMSON	☒ RNUMSON	☒ SNAPTOPIN
☒ SORT	☒ TEXT	☒ UNDO	☐ UNIQUE_LABEL
☒ VALUES	☒ XTRAERRS		

Figure 3-8 PROcapture Parameters Dialog Box

Consult your PRO Series documentation from Viewlogic or click on the Help button for an explanation of these options.

2. To activate the new settings, click on **`OK`**. To save them to the viewdraw.ini file, click on **`Save to Ini`**.

To find out the current settings of the key parameters without bringing up the dialog box, click on **`Info`** → **`Status`**. PROcapture brings up a window displaying the settings.

Navigating in PROcapture

This section describes the mouse buttons, keyboard commands, function keys, toolbar icons, menus, and dialog boxes available in PROcapture. It also tells you how to obtain help from the help screen, the Help toolbar icon, and the Help button in the dialog boxes.

Mouse Buttons

Mouse buttons perform the following functions in PRO Series:

- The left mouse button selects objects in PROcapture.
- The right mouse button cancels the current command mode. In addition, you use it to select multiple items; select the first item with the left mouse button and subsequent items with the right mouse button.

Keyboard Commands

You can use the keyboard to issue PROcapture commands, either the full command or its keyboard abbreviation.

Function Keys

You can use the function keys labeled F1, F2, F3, and so forth on your keyboard to invoke particular commands in the PRO Series tools. By default, they are assigned the functions shown in Figure 3-9.

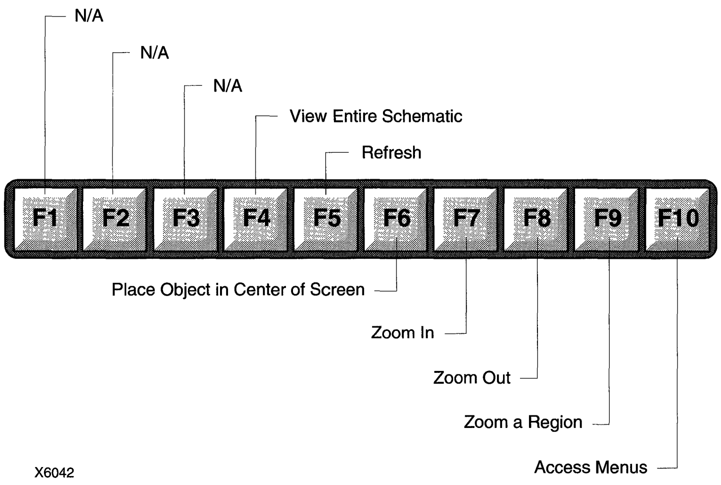


Figure 3-9 Default Function Keys

Toolbar Icons

Toolbar icons appear along the left-hand side of the PROcapture screen and in the toolbar underneath the menu commands when you load a design or open a blank schematic sheet. When you move the cursor over each icon, a description of its function appears in the status bar at the bottom of the PROcapture screen. See the "Toolbar Icons" section of this chapter for a description of most of the icons on the toolbar.

You can determine which icons are displayed in the toolbar of the PROcapture window.

1. Click on `View` → `Toolbar`.

The Toolbar dialog box appears, as shown in Figure 3-10.

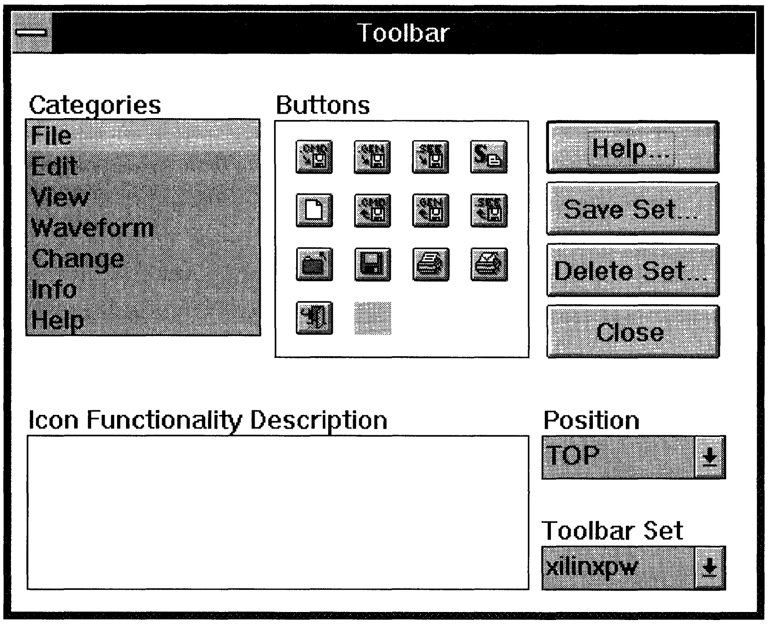


Figure 3-10 Toolbar Dialog Box

2. In the **Categories** field, select the type of icon to display in the Buttons field. For example, if you select **File**, only the icons that govern files, such as the Print icon or the Open icon, appear in the Buttons box.
3. You can add an icon to the toolbar by clicking on the icon in the **Buttons** field and dragging it to the toolbar. Similarly, you can delete an icon from the toolbar by dragging it into the **Buttons** field. When you click on an icon in this field, a brief description of its function appears in the Icon Functionality Description field.
4. Select **Save Set** to save the configuration of icons in the toolbar that you arranged by dragging icons from the Buttons field.
5. To load the set of icons that you saved with the Save Set command, click on **Toolbar Set**.

To remove the set of icons saved with the Save Set option, select **Delete Set**.

You can place the toolbar on the top, bottom, left-hand side, or right-hand side of the screen. Select Position and use the down arrow in the scroll bar to change this placement.

Menus

PROcapture offers nine menus. A description of each command on these menus is given in the "PROcapture Commands" chapter of this manual or in the PRO Series documentation from Viewlogic.

You can select menu commands with the mouse or the keyboard. With the mouse, click the left mouse button on the desired command. With the keyboard, press the Alt key and type the letter underlined in the command.

Dialog Boxes

An ellipsis (...) following a PROcapture menu command indicates that the command brings up a dialog box in which you can enter information and set options.

The following fields are common to many of the PROcapture dialog boxes:

- OK closes the dialog box and implements the intended action according to the settings in the dialog box.
- Cancel closes the dialog box without effecting any action.
- Help gives you information on how to use the dialog box.

Obtaining Help

You can obtain help on the PROcapture's commands and procedures by selecting commands on the Help menu, selecting the Help icon in the toolbar, or entering keyboard commands. In addition, the dialog boxes associated with some commands offer a Help button that you can click on to obtain context-sensitive help.

From the Toolbar

To obtain help from the toolbar, follow these steps.

1. Click on the Help toolbar icon, shown in Figure 3-11.

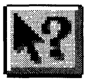
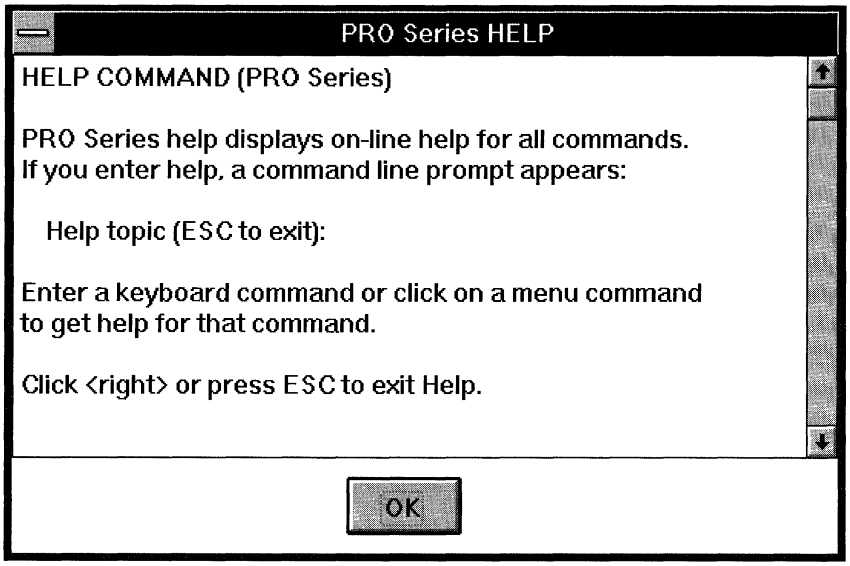


Figure 3-11 Help Toolbar Icon

The PRO Series Help screen appears, as Figure 3-12 indicates.



HELP COMMAND (PRO Series)

PRO Series help displays on-line help for all commands.
If you enter help, a command line prompt appears:

Help topic (ESC to exit):

Enter a keyboard command or click on a menu command
to get help for that command.

Click <right> or press ESC to exit Help.

Figure 3-12 PRO Series Help Screen

2. Click on **OK**, then select a menu command, or type in a keyboard command followed by a carriage return.

   The Help window for that command appears.

3. When you have read its contents, click **OK**, then select another command that you need help on, or press the Escape key to exit help mode.

From the Menu

Use the following procedure to obtain help from the menu using the mouse.

1. Select **`Help`** → **`Help`**.
2. Click on **`OK`** on the PRO Series Help screen, shown in Figure 3-12, then select the command for which you want help by either selecting the command from the menu or entering the keyboard equivalent from the keyboard.
3. When you have read its contents, click **`OK`**, then select another command that you need help on, or press the Escape key to exit help mode.

From the Keyboard

To obtain help from the keyboard, enter **`help`**↵ or **`help`** *command_name*↵ from the keyboard. Entering **`help`** *command_name*↵ brings up the help screen for that command; if you enter **`help`**↵, you must specify a command from either the menu or the keyboard.

Online Product Information

You can obtain online product information on PRO Series by clicking on **`Help`** → **`PROhelp`**. The following categories appear:

- Release Notes
- Installation Procedures
- Software Problem Reports
- Technical Reference Information
- Technical Support Services
- Product Data Sheets/Pricing

Click on any of these topics to obtain further information.

Moving Around the Screen

PROcapture works like any other Windows program; it allows you to view and work on several different designs simultaneously. The concept of working in several documents at once is known as the Multiple Document Interface, or MDI. MDI enables you to bring up multiple schematic windows and arrange them in the workspace in any fashion.

PROcapture also lets you zoom and pan around a schematic. Zooming allows you to view an entire schematic or focus in on a particular section. You can select the zoom commands from the View menu, from the toolbar, or by pressing the designated function key.

You can also make icons of schematics when the schematics are not needed. Making an icon of a schematic does not close the schematic; it merely turns the window into an icon. Later, when you need the schematic, you can double-click on the icon to re-display it.

Panning

You can familiarize yourself with a schematic by panning around the window. Panning is the process of obtaining a panoramic view of the screen by using the selected point as the center of the view.

To pan across the screen, first move the cursor to the location that will become the center of the new view, then press the **`F6`** function key. This step centers the edit area around the location of the cursor but does not move the cursor. Repeat moving the cursor and pressing F6 to move around the screen.

Zooming

Zooming magnifies or shrinks the view onscreen. You can either view the entire schematic or focus on one portion of it. All of the zoom commands are dynamic, which means you can use them while you are in the middle of another command.

- Click on the up arrow, shown in Figure 3-13, in the upper right corner of the schematic window to fill the entire workspace with the schematic window.

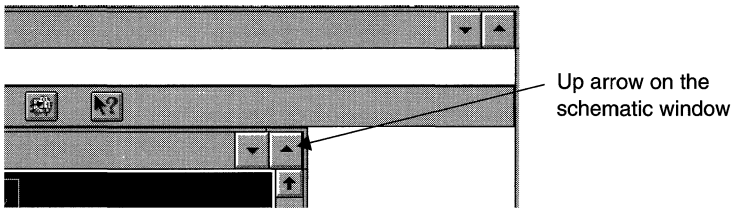


Figure 3-13 Up Arrow on the CALC.1 Schematic Window

Notice that when the window is enlarged to fill the work space, the schematic does not fill the entire area of the new schematic window.

- To view the entire sheet — that is, to expand the schematic to fill the entire window — select the **`View`** → **`Full`** command.

  Keyboard Shortcut: You can execute the View → Full command by pressing the **`F4`** function key.

  Toolbar Shortcut: You can execute the View → Full command by clicking on the Full toolbar icon, shown in Figure 3-14.

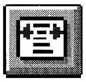


Figure 3-14 Full Toolbar Icon

- To zoom in, or magnify the view of the design, select the **`View`** → **`In`** command.

  Keyboard Shortcut: You can execute the View → In command by pressing the **`F7`** function key.

  Toolbar Shortcut: You can execute the View → In command by clicking on the In toolbar icon, shown in Figure 3-15.

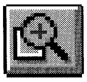


Figure 3-15 In Toolbar Icon

- To zoom out, or shrink the view of the design, select the **`View` → `Out`** command.

  Keyboard Shortcut: You can execute the View → Out command by pressing the **`F8`** function key.

  Toolbar Shortcut: You can execute the View → Out command by clicking on the Out toolbar icon, shown in Figure 3-16.

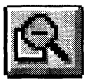


Figure 3-16 Out Toolbar Icon

- To zoom a particular region, select the **`View` → `Region`** command. Define the area to be zoomed by pressing and holding the left mouse button in the upper left corner of the area and the left mouse button in the lower right corner. Release the left mouse button.

  Keyboard Shortcut: You can execute the View → Region command by pressing the **`F9`** function key. You define the upper left corner when you select the command and the lower right corner when you press the left mouse button.

- To zoom a particular object, select the object with the left mouse button, and click on **`View` → `Selected`**. PROcapture magnifies the object and places it at the center of the view.

Making Icons of Schematics and Symbols

Use the following procedure to make an icon of the current window.

1. When the current window occupies the entire screen, the name of the schematic is shown in the PROcapture title bar. You can reduce the size of the window by pressing the up/down arrow button to the right of the Help menu.

2. To make an icon of the current window, press the down arrow button in the upper right of the window. Figure 3-17 illustrates the resulting icon.

   To re-open the window, double-click on the icon.

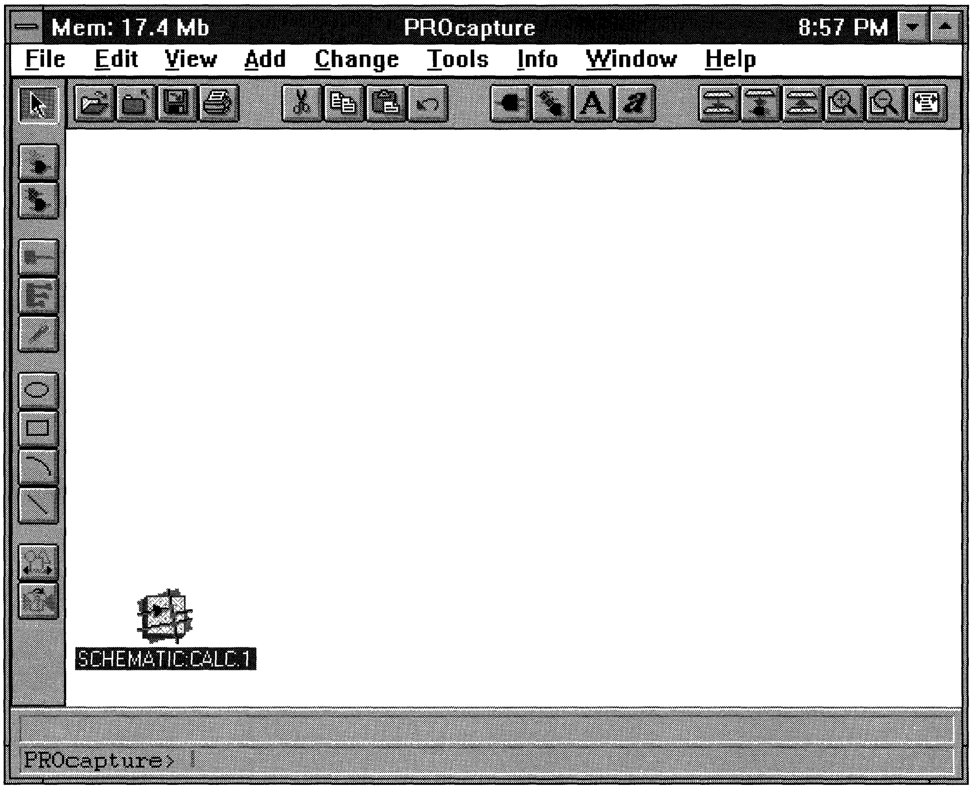


Figure 3-17 Schematic Icon

Undoing Commands

You can undo the execution of the most recent command by clicking on **`Edit`** → **`Undo`**.

Toolbar Shortcut: You can execute the Edit → Undo command by clicking on the Undo toolbar icon, shown in Figure 3-18.

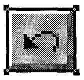


Figure 3-18 Undo Toolbar Icon

Adding Components

Follow these steps to place library components on your schematic.

1. To place a component on the schematic, select the **`Add`** → **`Component`** command.

   Mouse Shortcut: You can execute the Add → Component command by double-clicking the left mouse button in an unused area of the schematic.

Note: If double-clicking the left mouse button does not display the Add Component dialog box, verify that the DBOXON box is checked in the PROcapture Parameters dialog box.

   Keyboard Shortcut: You can execute the Add → Component command by typing **`com`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Component command by clicking on the Component toolbar icon, shown in Figure 3-19.

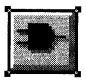


Figure 3-19 Component Toolbar Icon

   The Add → Component command brings up the Add Component dialog box, shown in Figure 3-20.

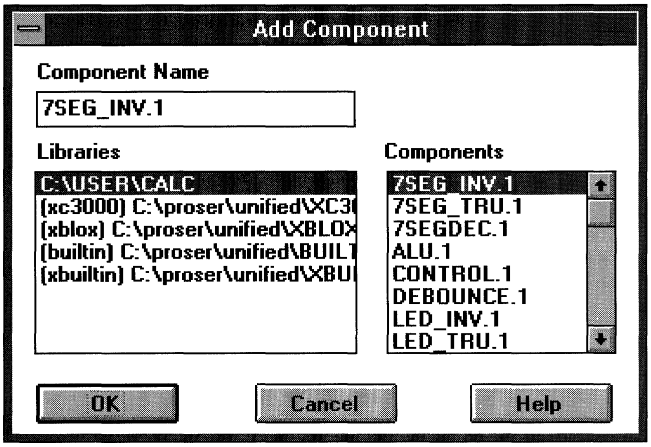


Figure 3-20 Add Component Dialog Box

When you initially bring up the Add Component dialog box, PROcapture displays the first component in the selected library in the Component Name field. When you change the selected library in the Libraries list box, the available components of that library are displayed in the Components list box.

2. To view the available components in a library, select the library in the **`Libraries`** list box.

   The Components list box in the dialog box is now updated.

3. Using the down arrow in the scroll bar of the **`Components`** list box, scroll down until the desired component is displayed.

Note: Once the Components list box is selected, you can jump to a component by typing the first character of the component's name.

4. Select the component.

   The component is now highlighted, and the Component Name field is updated.

5. Click on **OK**.

   The Add Component dialog box closes and the schematic window is reactivated. An outline of the component being added appears at the end of the pointer.

Note: If you want to see the entire component instead of an outline, check the DETAIL box in the PROcapture Parameters dialog box.

6. To place the component, click the left mouse button on the desired location in the schematic window.

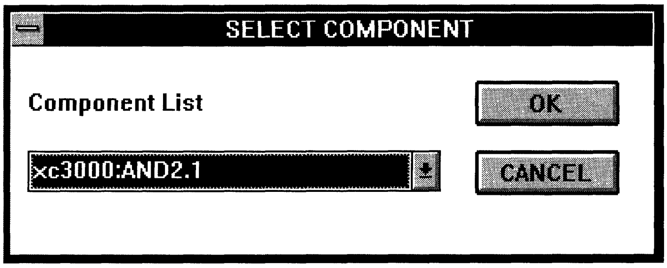
Changing Components

Rather than placing a new component, you can convert a placed component to another component. This method is useful if you have placed the wrong component and want to replace it with the correct component.

1. Select the **`Edit`** → **`Select`** → **`Component`** command.

   Keyboard Shortcut: You can execute the Edit → Select → Component command by typing **`sco`**↵ on the PROcapture command line. You can also specify the component to select. For example, to select all AND2 components, type sco and2↵.

   The Select Component dialog box appears, as indicated in Figure 3-21. It displays all the components in the active schematic. By default, the last component selected appears in the Component List field. Scroll down until you find the component that you want to change.



xc3000:AND2.1

Figure 3-21 Select Component Dialog Box

2. Click on **OK**.

   The Select Component dialog box closes and the schematic is reactivated with all the chosen components selected.

Note: This procedure selects *all* the components of the same type, for example, all AND2 components. To change only a single component, click on the component.

3. Select the **`Change`** → **`Component`** command.

   Keyboard Shortcut: You can execute the Change → Component command by typing **cc**↵ on the PROcapture command line. You can also specify the component to change to. For example, to change all selected components to an OR2, type cc or2↵.

   The Change → Component command brings up the Change Component dialog box, shown in Figure 3-22.

4. In the **`Libraries`** field, select the library in which the component that you want to change to resides.

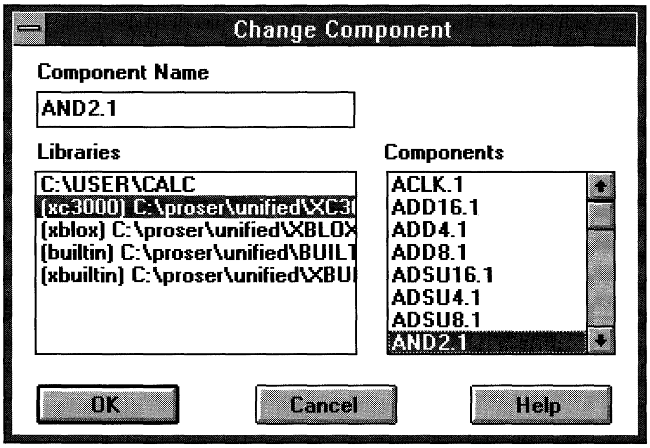


Figure 3-22 Change Component Dialog Box

5. In the **`Components`** field, use the down arrow in the scroll bar to scroll down until the desired component is displayed.

6. Select the desired component.

   The component is highlighted, and the Component Name field is updated, as indicated in Figure 3-23.

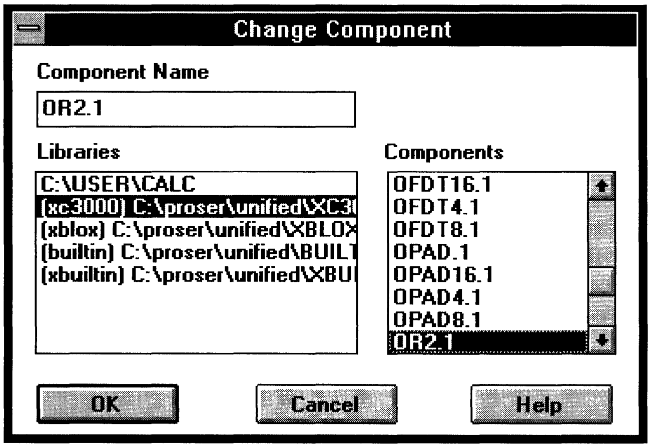


Figure 3-23 Updated Component Name

7. Click on **OK**.

   Selecting OK closes the Change Component dialog box and updates the schematic.

Adding Nets

Nets and buses establish connectivity between pins on the same hierarchical level of a design. However, it is not necessary to physically connect nets on the schematic. If two dangling nets or buses within a single schematic are labeled with the same name, they are considered electrically connected. Labeling unconnected nets with the same name is sometimes a useful technique that can make schematics easier to read, especially when dealing with clocks. However, you must keep track of all signal names and make sure they match exactly.

Note: Net name association applies to nets on different schematic sheets. For example, a net on design.1 named DATA0 is electrically connected to another net on design.2 named DATA0.

Use the following procedure to add a net.

1. Select the **Add** → **Net** command.

Keyboard Shortcut: You can execute the Add → Net command by typing **`net`**↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the Add → Net command by clicking on the Net toolbar icon, shown in Figure 3-24.

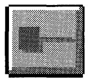


Figure 3-24 Net Toolbar Icon

Selecting Add → Net changes the pointer to a crosshair and enters Add Net mode, which is reflected in the PROcapture command line.

2. Point the crosshair at the point where the net is to start. This point must be on a net, bus, or pin.
3. Click the left mouse button.
4. Click the left mouse button each time that you want the net to pivot.
5. Move the mouse to the desired end point of the net and click the left mouse button.
6. When creating a dangling net, which is a net that does not end at a net, bus, or pin, click the right mouse button to complete the net.

Adding Buses

Sometimes it is convenient to draw a set of signals as a bus rather than as several separate wires. It is not necessary to connect a bus physically with the nets that make up the bus.

To add a bus, follow these steps.

1. Select the **`Add`** → **`Bus`** command.

   Keyboard Shortcut: You can execute the Add → Bus command by typing **`bus`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Bus command by clicking on the Bus toolbar icon, as shown in Figure 3-25.

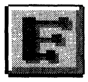


Figure 3-25 Bus Toolbar Icon

Selecting Add → Bus changes the pointer to a crosshair and enters Add Bus mode. The mode is reflected in the PROcapture command line.

2. Point the crosshair at the point where the bus is to start.
3. Click the left mouse button to begin the bus.
4. Click the left mouse button each time that you want the bus to pivot.
5. Move the mouse to the desired end point of the bus and click the left mouse button.
6. To create a dangling bus, which is a bus that does not end at a net, bus, or pin, click the right mouse button to complete the bus.

Adding Arrays

It is sometimes convenient to create an array of objects rather than adding each object individually. To add an array, follow these steps.

1. Select the object from which to create the array.
2. Click on **`Add`** → **`Array`**.

   Keyboard Shortcut: You can execute the Add → Array command by typing **`array`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Array command by clicking on the Array toolbar icon, as shown in Figure 3-26.

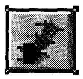


Figure 3-26 Array Toolbar Icon

3. At the `Number of Rows [5]:` prompt, enter the number of rows in the array; the default value is shown between the square brackets.
4. At the `Number of Columns [3]:` prompt, enter the number of columns in the array; the default value is shown between the square brackets.
5. At the `Rel/Abs Spacing(r/a) [r]:` prompt, enter **r** for relative spacing or **a** for absolute spacing.
    - In relative spacing, the column and row spacing is measured from the border of the bounding box surrounding the selected object.
    - In absolute spacing, the column and row spacing is measured from the lower left corner of the bounding box that surrounds the selected object. The objects in the array sometimes overlap when you select absolute spacing.
6. At the `Column spacing [10]:` prompt, enter the number of grid units by which the columns should be spaced.

    Specify a positive column spacing to spread the columns from left to right. You can use a negative number to spread the columns from right to left.
7. At the `Row spacing [10]:` prompt, enter the number of grid units by which the columns should be spaced.

    Specify a positive row spacing to spread the rows from bottom to top. You can specify a negative number to spread the rows from top to bottom.

Adding Graphical Objects

You may want to add a box to a symbol or a schematic, or add graphical objects to the screen for annotational purposes. This section describes how to add these objects to your schematic. Like text, graphical objects are visual only.

Adding Lines

Follow this procedure to add a line.

1. Click on **`Add`** → **`Line`**.

   Keyboard Shortcut: You can execute the Add → Line command by typing **`line`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Line command by clicking on the Line toolbar icon, shown in Figure 3-27.

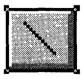


Figure 3-27 Line Toolbar Icon

2. Point the crosshair at the location to start the line, and click on the left mouse button.
3. To pivot the line in another direction, click the left mouse button.
4. To end the line or lines, click the right mouse button.
5. Press Escape, click the right mouse button, or click on the Clear toolbar icon, shown in Figure 3-28, to terminate this command.

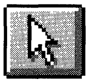


Figure 3-28 Clear Toolbar Icon

Adding Boxes

To add a box, follow these steps.

1. Click on **`Add`** → **`Box`**.

   Keyboard Shortcut: You can execute the Add → Box command by typing **`box`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Box command by clicking on the Box toolbar icon, shown in Figure 3-29.

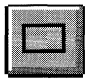


Figure 3-29 Box Toolbar Icon

2. Point the crosshair at the point to start the box and hold down the left mouse button.
3. Keeping the left mouse button pressed, draw the box.
4. Release the left mouse button.
5. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

When you create a box, the status line gives the dimensions of the box. The Box toolbar icon remains selected after the box has been defined because PROcapture remains in Add Box mode. You can add multiple boxes in succession without having to re-invoke the Add → Box command.

Adding Arcs

Use the following procedure to add an arc.

1. Click on **`Add`** → **`Arc`**.

   Keyboard Shortcut: You can execute the Add → Arc command by typing **`arc`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Arc command by clicking on the Arc toolbar icon, shown in Figure 3-30.

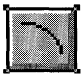


Figure 3-30 Arc Toolbar Icon

2. Point the crosshair at the location of the arc's first end point.
3. While holding down the left mouse button, point the crosshair at the location of the arc's second end point.

4. Release the left mouse button and move the mouse until the arc attains the desired shape.
5. Press the left mouse button to complete the arc.
6. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Adding Circles

Follow these instructions to add a circle.

1. Click on **`Add`** → **`Circle`**.

   Keyboard Shortcut: You can execute the Add → Circle command by typing **`circle`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Circle command by clicking on the Circle toolbar icon, shown in Figure 3-31.

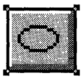


Figure 3-31 Circle Toolbar Icon

2. Point the crosshair at the location of the center of the circle.
3. While holding down the left mouse button, draw the circle.
4. Release the left mouse button.
5. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Adding Color and Pattern to Objects

Follow this procedure to add color and pattern to objects.

1. Select the object to be modified with the left mouse button.
2. Select **`Change`** → **`Object Color`** to change the object's color or **`Change`** → **`Fill Style`** to fill it with a pattern.

   Keyboard Shortcut: You can execute the Change → Object Color command by typing **`col`**↵ on the PROcapture command line.

Keyboard Shortcut: You can execute the Change → Fill Style command by typing **`fs`**↵ on the PROcapture command line.

A bar of colors or fill patterns, respectively, appears along the bottom of the screen.

3. Click on the desired color or pattern with the left mouse button.

   The object is now filled with the selected color or pattern.

4. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Rotating Objects

You can rotate an object 90 degrees counterclockwise from its present position.

1. Select the object to be rotated with the left mouse button.
2. Click on **`Change`** → **`Transform`** → **`Rotate`**.

   Keyboard Shortcut: You can execute the Change → Transform → Rotate command by typing **`ro`**↵ on the PROcapture command line.

   The object now appears in its new orientation.

3. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Mirroring Objects

You can reflect an object through an axis.

1. Select the object to be mirrored with the left mouse button.
2. Click on **`Change`** → **`Transform`** → **`Reflect`**.

   Keyboard Shortcut: You can execute the Change → Transform → Reflect command by typing **`refl`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Change → Transform → Reflect command by clicking on the Reflect toolbar icon, shown in Figure 3-32.

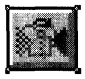


Figure 3-32 Reflect Toolbar Icon

3. While holding down the left mouse button, draw a temporary reflection axis to reflect the object through.

   The object now flips to the other side of the line and reverses its orientation. The temporary axis disappears.

4. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Scaling Objects

You can scale objects, which changes their size while keeping them proportional to the original size.

1. Select the object to be scaled with the left mouse button.
2. Click on **`Change`** → **`Transform`** → **`Scale`**.

   Keyboard Shortcut: You can execute the Change → Transform → Scale command by typing **`sca`**↵ on the PROcapture command line.

3. On the command line, enter the scale factor by which to multiply the current size. To make the object larger, the factor must be greater than 1; to make it smaller, the factor must be less than 1. For example, entering a factor of .50 reduces the size of the object to half of its original size; conversely, entering a factor of 2 doubles its size.
4. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Re-Shaping Objects

You can change the size and shape of objects.

1. Select the object to be reshaped with the left mouse button.
2. Click on **`Change`** → **`Transform`** → **`Stretch`**.

Keyboard Shortcut: You can execute the Change → Transform → Stretch command by typing `stre`↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the Change → Transform → Stretch command by clicking on the Stretch toolbar icon, shown in Figure 3-33.

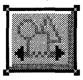


Figure 3-33 Stretch Toolbar Icon

3. Place the crosshair on the part of the object that you want to enlarge.
4. While holding down the left mouse button, drag the cursor until the object assumes the desired shape and release the mouse button.
5. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

User-Defined Macros

A macro is any symbol defined by an underlying Viewlogic schematic. The Xilinx libraries contain several symbols that are macros. Macro schematics contain primitives and/or other macro symbols. When the software reads a schematic, it expands (flattens) the macro symbols into their underlying schematics. The components actually processed and reported by the software are the underlying primitives, referenced by their hierarchical instance names, after macro expansion.

You can create your own custom macro symbols to use in your designs. The procedure for creating a custom macro is the same for EPLDs and FPGAs. You can create and store your custom macro symbols and their underlying schematics in your local project directory, or you can create a custom library directory to store your custom macros so they can be used in multiple projects. See the "Design and Simulation Techniques" chapter for instructions on creating a custom library. You should never add custom symbols or macros to the Xilinx library directories or modify any of the library symbols or macros. You can, however, copy any of the Xilinx-supplied macros from the library into your project or custom library directory and modify them to suit your design needs, saving the modified component to your local primary directory.

Note: When using a Xilinx library macro as a template for a new component, it is good practice to specify a new name when saving the component.

When you create a custom macro symbol, the Viewlogic block type must be set to Composite, not Module. No symbol attributes are required by the Xilinx software for macro symbols. However, if you copy a Xilinx-supplied primitive symbol from the library to use as the basis of your custom macro symbol, make sure you delete the LEVEL=XILINX symbol attribute, because this attribute marks the symbol as a primitive.

Note: If the LEVEL=XILINX attribute is not displayed when you edit the symbol, type **`avis`** *↵ to display all attributes.

Creating Symbols (Macros)

A symbol is a graphic representation of a level of hierarchy. Symbols can represent user-defined macros or design files from other sources. This section describes how to create a symbol.

1. Open a blank symbol window by clicking on **`File`** → **`Open`**, which displays the File Open dialog box.

   Keyboard Shortcut: You can execute the File → Open command for a symbol by typing **`sym`**↵ on the PROcapture command line. You can also specify the symbol to be opened. For example, to open a symbol named ALU, type sym andblk2↵.

2. In the **`Design Name`** field, type in the symbol name.

3. Select the **`Symbol`** setting in the Design Type field. The default setting is Schematic.

4. Click on **`OK`**.

   The File Open dialog box closes and a symbol window opens.

The symbol window contains a box called a block sheet. It defines the perimeter of the symbol but does not show up on the screen when the symbol is placed in a design schematic. Only the elements that you add to the symbol are visible.

The initial size of the symbol, shown as the area defined by the block sheet, is 1 inch by 1 inch or 100 x 100 grid units. For most symbols, it is necessary to enlarge or reduce this default size.

Changing Symbol Size

To change the size of the symbol, follow these instructions.

1. While editing the symbol, select the **`Change`** → **`Sheet Size`** → **`Z-WxH`** command.

   PROcapture prompts you first for the new block width at the PROcapture command line.

2. At the `Block width [100]` prompt, type in the desired symbol width, and press ↵.

   Now PROcapture prompts you for the new block height.

3. At the `Block height [100]` prompt, type in the desired symbol height, and press ↵.

When you change the size of a symbol, the change is reflected in the text at the top of the symbol window. An example is the following:

```
ANDBLK2.1(SYM) Z-1.5"x1.2" G:10
```

This text tells you that the window is for a symbol called ANDBLK2.1, which has the dimensions of 1.5 inches by 1.2 inches. The G:10 notation indicates that the grid spacing is every tenth of an inch. You can change the grid spacing to any desired value, but 10 works well with the Unified Libraries because the pins on the various components are spaced by multiples of 10.

Creating Symbol Box

Most symbols have a visible frame or symbol body to which its pins are attached. To add a box to a symbol, follow the steps outlined in the "Adding Boxes" section earlier in this chapter.

Adding Pins

To add pins to the symbol, follow these steps.

1. Select the **`Add`** → **`Pin`** command.

   The pointer changes to a crosshair.

   Keyboard Shortcut: You can execute the Add → Pin command by typing **`pin`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Add → Pin command by clicking on the Pin toolbar icon, shown in Figure 3-34.

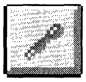


Figure 3-34 Pin Toolbar Icon

2. On the symbol body, point the crosshair at the spot where you want the pin to start.
3. While holding down the left mouse button, drag the mouse to the spot on the block sheet boundary where you want the pin to end.
4. Release the left mouse button.

   When a pin is created, the status line gives the dimensions of the pin. The Pin toolbar icon remains depressed after the box has been defined because PROcapture remains in Add Pin mode. You can add multiple pins in succession without having to re-invoke the Add → Pin command.
5. Press Escape, click the right mouse button, or click on the Clear toolbar icon to terminate this command.

Adding Pin Labels

Pin labels must exactly match the labels used for the same signals in the corresponding schematic. For example, if there is a pin labeled "clock" on the symbol, there must be a net labeled "clock" in the symbol's schematic. To add a pin label, follow the procedure given in the "Adding Labels" section.

Adding Pin Attributes

You can attach attributes to pins as well as to symbols. The most common attribute applied to a pin is the PINTYPE attribute. The valid values for the PINTYPE attribute are IN, OUT, and BI. PINTYPE attributes are optional except when there is no schematic for a symbol. (See the "Merging Non-Schematic-Based Modules" section of the "Design and Simulation Techniques" chapter for a discussion of this special case.) You can add attributes to pins using the procedure given in the "Adding Attributes" section later in this chapter.

Merging Design Files from Other Sources

You can enter part of your design in some form other than schematics, such as state machine entry or a RAM or ROM description. You can also bring in netlist files produced by interface software from a Xilinx Alliance partner. Whatever the form of entry, the starting point for inclusion into a Viewlogic schematic design must be a Xilinx Netlist Format (XNF) file. *This file must be located in the project directory.* Without an XNF file, this portion of the design cannot be included; with it, the origin of the logic becomes irrelevant.

As an exception, you do not need to convert the PLUSASM files for EPLD designs to XNF format; the implementation software reads them in.

See the "Design and Simulation Techniques" chapter for instructions on creating custom symbols for non-schematic-based modules.

Note: The pins of the created symbol must have PINTYPE attributes specifying the directionality of the module pins.

Adding Labels

Labeling is the process of identifying a net, bus, component, or pin by assigning a text string to it. It is strongly recommended that you label all nets and buses on the schematic to make debugging easier.

Follow the conventions described in the following sections when you add labels.

Naming Conventions

FPGA names for nets, buses, components, and pins must follow these conventions:

- Only A-Z, a-z, 0-9, "_," and "-" are allowed in user-defined names. No other characters should be included in names.
- Names must contain at least one non-numeric character.
- Names cannot be more than 1024 characters long.

Do not use a hyphen (-) as the first character of a signal or a symbol name. The hyphen character has a special meaning to PROsim, the Viewlogic simulator; a hyphen can cause problems in specifying a net name during simulation.

The translation programs replace invalid characters found in a given name with an underscore (_); however, a tilde (~) is replaced with a hyphen (-).

Note: Do not use the Change → Object Label → Set Sense command in PROcapture to add an overscore to a label, because it causes problems during simulation and debugging. In translation, the overscore is changed to a tilde (~) in the WIR file, and the Xilinx software changes the tilde to a hyphen (-).

Reserved Names

The physical names associated with every resource on every part are reserved and cannot be used to name signals and symbols. These include CLBs, IOBs, clock buffers, BUFTs, oscillators, package pin names, CCLK, DP, GND, VCC, RT, PWRDN, and RST. Other examples are CLB names such as AA and AB, pin names such as P1 and P2, pad names such as PAD1 and PAD2, and primitive names such as TDO, BSCAN, M0, M1, M2, or STARTUP.

Net Names

Hierarchical signal names are fully specified in the XNF file. Here are some examples:

- Unlabeled signals are given internal names generated automatically by PROcapture that consist of a dollar sign, sheet number, "N" for net or bus, and a unique number assigned by PROcapture for each net. Any change to the schematic changes at least some of the PROcapture-assigned internal names.

- ABC represents a labeled signal named ABC in the top-level drawing.
- $1I11\ABC represents a labeled signal named ABC that is underneath an unlabeled component called $1I11, where $1I11 is the symbol reference designator named with a dollar sign. It is derived from the sheet number (sheet 1), "I" for instance, and a unique instance number assigned by PROcapture after each instance, in this case, "11."
- $1N118 represents net $1N118, which is on sheet 1 of the root-level drawing.
- $1I5\$1N118 represents net $1N118, which is a net within the top-level symbol $1I5. If the net is within a schematic represented by a symbol on the design's top level, the default signal name reflects this hierarchy.

As these examples clearly show, the more labels you put in your design, the easier you will find it to locate signals for simulation. For more information on adding labels to nets, see the Add → Label command in the "PROcapture Commands" appendix.

Component Names

To give components more meaningful names than those issued by PROcapture, use the Add → Object Label command to name symbols, just as you would nets. The following are examples of symbol names.

- MYSYM is a component located at the top level of the drawing.
- TOP\MYSYM is a component located one level below TOP.

Components with or without user-assigned labels are assigned names as follows:

top-level_instance\instance

For example, $1I3\$1I5 represents a component (instance I5) located one level below symbol $1I3.

Bus Names

To ensure that bus signals are processed correctly, use the following naming conventions.

- All buses and all nets going into buses must be labeled. For example, a bus labeled A[0:2] should have nets labeled A0, A1, and A2.
- The WIR2XNF program expands bus notation. All bus and symbol pin names are expanded into individual signal or pin names. For example, a bus labeled DATA[0:3] is converted into four nets labeled DATA0, DATA1, DATA2, and DATA3.

You must be consistent in the order of bus indices for a single bus. For example, do not connect busa[0:3] to busb[3:0] at another level of your schematic unless you are deliberately reversing the bus order.

See Table 3-1 for examples of legal bus names.

Table 3-1 Bus Name Examples

Bus Name	Description
Q[0:7]	8-bit bus, signals Q0 (MSB) through Q7 (LSB)
Q[7:0]	8-bit bus, signals Q7 (MSB) through Q0 (LSB)
Q[7:0],SET,CLK	10-bit bus, signals Q7 through Q0; also signals SET and CLK
A[7:0],B[7:0]	16-bit bus, signals A7 through A0 and signals B7 through B0
DATA[0:7:2]	4-bit bus, signals DATA0, DATA2, DATA4, and DATA6
DATA[0:F/H]	16-bit bus, specified in hexadecimal (You can also specify a bus in decimal, octal, or binary.)

Adding a Label

Follow these steps to add a label to a net, bus, component, or pin.

1. Using the left mouse button, select the object to be labeled.

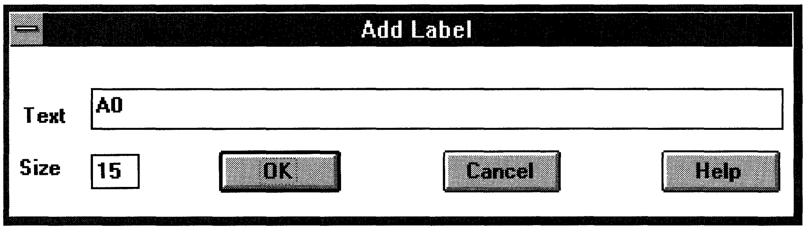
2. Select the **`Add → Object Label`** command.

   Mouse Shortcut: You can execute the Add → Object Label command on an object by double-clicking on the object to which you want to add the label.

   Keyboard Shortcut: You can execute the Add → Object Label command by typing **`la`**↵ on the PROcapture command line.

   The Add Label dialog box appears.

3. Type the name of the object in the **`Text`** field of the dialog box, as indicated in Figure 3-35.



Text: A0
Size: 15

Figure 3-35 Text in the Add Label Dialog Box

4. Click on **`OK`**.

   The Add Label dialog box closes, and the current window is reactivated. An outline of the label, or bounding box, appears on the screen next to the pointer.

5. Point the mouse so that the label outline is in its desired location.

6. Click the left mouse button to place the label.

All buses and nets going into a bus must be labeled. See the "Net Names," "Component Names," and "Bus Names" sections in this chapter for information on how to label these entities correctly.

Changing Label Text

If you mislabel a net or bus, you do not have to delete the label. To change the string, select the label and choose the **`Change → Text`** command, or double-click on the text to bring up the Change Text dialog box, which is shown in Figure 3-36.

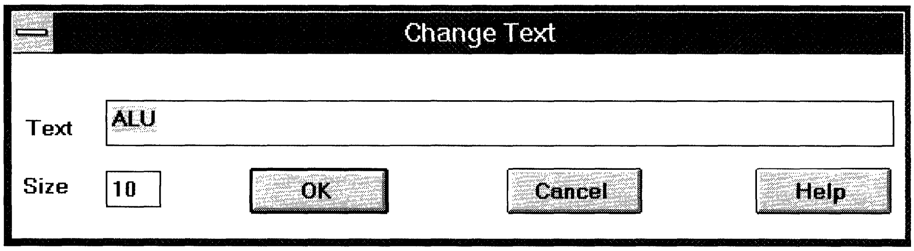


Figure 3-36 Change Text Dialog Box

Changing Label Visibility

Labels can be either visible or invisible.

Making Labels Invisible

To make a label invisible, follow these steps.

1. Select the **`Change`** → **`Object Label`** → **`All Labels Off`** command.

   On the command line, you see a `Label text string [*]:` prompt.

2. Type in the names of the labels to be turned off. If you press ↵, all labels are turned off.

Making Labels Visible

To make a label visible, follow these steps.

1. Select the **`Change`** → **`Object Label`** → **`All Labels On`** command.

   On the command line, you see a `Label text string [*]:` prompt.

2. Type in the names of the labels to be turned on. If you press ↵, all labels are turned on.

Adding Text

To distinguish one symbol from another, or to add notes to a schematic, you can add text such as the component name or revision number to the symbol or schematic. Text is purely visual.

1. Select the **Add → Text** command to bring up the Add Text dialog box.

   Toolbar Shortcut: You can execute the Add → Text command by clicking on the Text toolbar icon, shown in Figure 3-37.

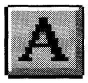
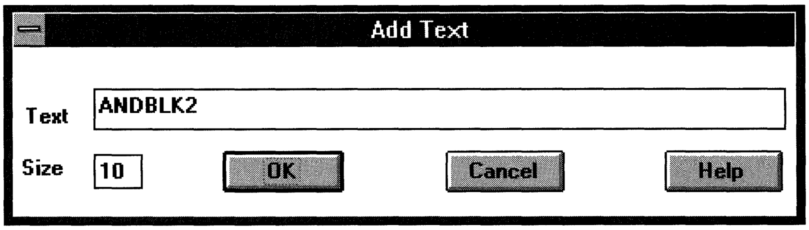


Figure 3-37 Text Toolbar Icon

2. In the **Text** field, type the desired text, as illustrated in Figure 3-38.



Text ANDBLK2

Size 10

Figure 3-38 Add Text Dialog Box

3. Click on **OK**.

   Selecting OK closes the Add Text dialog box and reactivates the symbol window. A bounding box is displayed next to the pointer.

4. Move the mouse so that the bounding box is in the desired position.
5. Click the left mouse button to place the text string.

If you make a mistake while typing the text, or want to change the text after you have placed it on the symbol, double-click on the added text. The Change Text dialog box comes up so that you can edit the text.

Changing Text Size

Depending on the size of the schematic sheet, any text, including attributes, that you have added to the component may not be visible at full zoom. It may be necessary to change the text size.

1. Double-click on the component or symbol text.

   The Change Text dialog box appears.

2. Change the size in the dialog box.
3. Click on **OK** to close the Change Text dialog box and update the selected text.

Adding Attributes

Attributes are instructions placed on symbols or nets in an FPGA or EPLD schematic to indicate their placement, implementation, naming, directionality, and so forth. The list of attributes that you can place on the components in your Viewlogic schematic is given in the *Xilinx Libraries Guide.*

To assign attributes to nets, buses, components, or pins, you can either use the Add → Object Attribute command or change an existing attribute.

Using the Add Object Attribute Command

One way to add an attribute is to use the Add → Object Attribute command.

1. Using the left mouse button, select the object to add the attribute to.

   Selecting the object changes its color and displays its bounding box.

2. Select the **Add → Object Attribute** command.

Keyboard Shortcut: You can execute the Add → Object Attribute command by typing **`at↵`** on the PROcapture command line.

3. At the `Attribute Text String:` prompt, type *attribute=attribute_name↵*, for example, PINTYPE=IN↵.

   When you enter the attribute, an outline of the attribute, or bounding box, appears on the screen next to the pointer.

4. Point the mouse so that the attribute outline is in its desired location.

5. Click the left mouse button to place the attribute.

Changing an Existing Attribute

Changing an existing attribute is another way to add an attribute.

1. Using the left mouse button, select the component or net.

2. Select the **`Change`** → **`Object Attributes`** → **`Dialog`** command.

   Toolbar Shortcut: You can execute the Change → Object Attributes → Dialog command by clicking on the Attribute toolbar icon, shown in Figure 3-39.

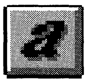


Figure 3-39 Attribute Toolbar Icon

Mouse Shortcut: You can execute the Change → Object Attributes → Dialog command by double-clicking on an attribute.

The Edit Attributes dialog box appears, as illustrated in Figure 3-40.

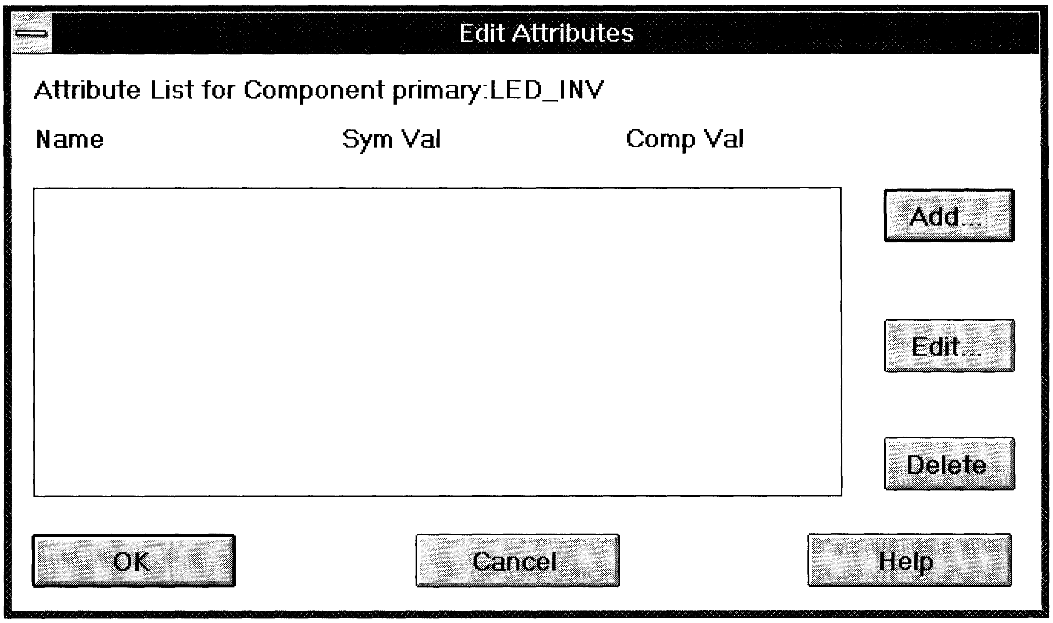


Figure 3-40 Edit Attributes Dialog Box

3. To add a new attribute, click on **Add**.

   Selecting Add displays an empty Edit Attribute dialog box.

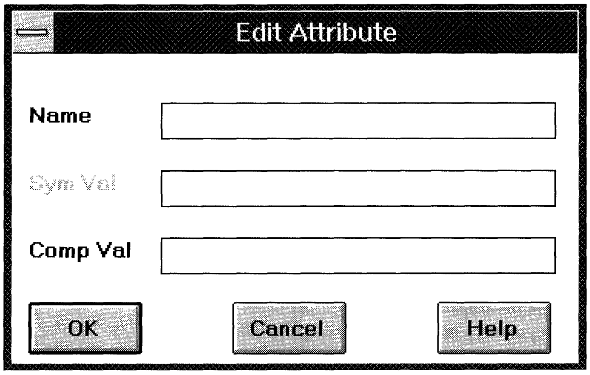
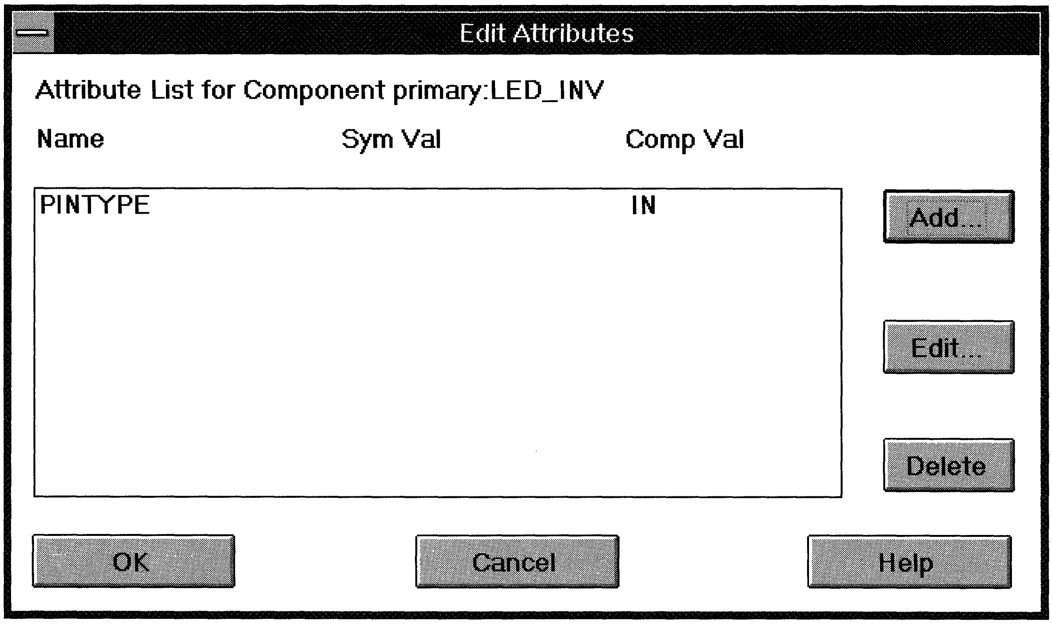


Figure 3-41 Edit Attribute Dialog Box

4. In the **Name** field, type the attribute name.
5. In the **Comp Val** field, type the attribute value.
6. Click on **OK**.

   The Edit Attributes dialog box now reflects the new attribute; Figure 3-42 gives an example.



Attribute List for Component primary:LED_INV

Name	Sym Val	Comp Val
PINTYPE		IN

Figure 3-42 Updated Edit Attributes Dialog Box

7. Click on **OK**.

   The Edit Attributes dialog box closes, and the schematic or symbol window is reactivated with the attribute added to the object.

Note: To edit an attribute that already exists on the object being modified, use the Edit button.

Changing Attribute Size

Once placed on an object, attributes can be modified in the same way as text. To change the size of an attribute, use the procedure outlined in the "Changing Text Size" section.

Changing Attribute Visibility

You can make attributes visible or invisible. Rendering the attributes invisible is for cosmetic reasons only; the attributes still affect the components or symbols.

Making Attributes Invisible

To make an attribute invisible, follow these steps.

1. Select the **`Change`** → **`Object Attributes`** → **`Visibility`** → **`All Attrs Off`** command.

   Keyboard Shortcut: You can execute the Change → Object Attributes → Visibility → All Attrs Off command by typing **`ain`**↵ on the PROcapture command line.

2. At the `Attribute Text String [*]:` prompt, press ↵.

   Pressing ↵ selects the default wildcard value, which turns all the symbol attributes invisible.

3. To redraw the screen, press **`F5`** or select **`View`** → **`Refresh`**.

Making Attributes Visible

To make an attribute visible, follow these steps.

1. Select the **`Change`** → **`Object Attributes`** → **`Visibility`** → **`All Attrs On`** command.

   Keyboard Shortcut: You can execute the Change → Object Attributes → Visibility → All Attrs On command by typing **`avi`**↵ on the PROcapture command line.

2. At the `Attribute Text String [*]:` prompt, press ↵.

   Pressing ↵ selects the default wildcard value, which turns all the symbol attributes visible.

3. To redraw the screen, press **`F5`** or select **`View`** → **`Refresh`**.

Copying Objects

Once a component, graphical object, or symbol is placed on the schematic, you can copy it. You can use either of two methods to copy the object.

Copying Directly to the Schematic

In this method, you use a toolbar icon, not a a menu command, to copy an object. This method does not copy objects to the clipboard.

1. If the object to be copied does not have a box surrounding it, select it by clicking the left mouse button.
2. Select the Copy toolbar icon, shown in Figure 3-43:

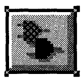


Figure 3-43 Copy Toolbar Icon

Keyboard Shortcut: You can execute the Copy command by typing **`copy`**↵ on the PROcapture command line.

Selecting the Copy toolbar icon changes the pointer to a crosshair and enters Copy mode, which is reflected in the PROcapture command line. If the pointer does not change to a crosshair, you have not selected the item to copy.

3. While holding down the left mouse button in the current window, drag the mouse to the desired location of the new object.
4. Release the left mouse button to place the copied object.

Copying to the Clipboard

The second method of copying uses a menu command that copies an object to the clipboard.

1. If the object to be copied does not have a box surrounding it, select it by clicking the left mouse button.
2. Select the **`Edit`** → **`Copy`** command.

Keyboard Shortcut: You can execute the Edit → Copy command by typing **bcop**↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the Edit → Copy command by clicking on the Edit Copy toolbar icon, shown in Figure 3-43.

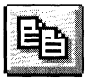


Figure 3-44 Edit Copy Toolbar Icon

3. Select the **`Edit`** → **`Paste`** command and move the box to the desired location.

   Keyboard Shortcut: You can execute the Edit → Paste command by typing **bpas**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Edit → Paste command by clicking on the Paste toolbar icon, shown in Figure 3-45.

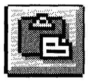


Figure 3-45 Paste Toolbar Icon

   Selecting Edit → Paste displays a bounding box of the components, nets, and buses that are being pasted on the screen next to the pointer.

4. Click the left mouse button to place the object.

Moving Objects and Text

Once an object such as text or a component is placed, you can easily move it.

Moving Objects

To move an object, follow these steps.

1. Select the component to be moved.

2. After selecting the first component, use the right mouse button to select any additional components to be moved.
3. Select the **`Edit`** → **`Move`** command.

   Mouse Shortcut: You can execute the Edit → Move command by selecting the component to be moved and dragging it to the new location.

   Keyboard Shortcut: You can execute the Edit → Move command by typing **`move`**↵ on the PROcapture command line.

   Toolbar Shortcut: You can execute the Edit → Move command by clicking on the Move toolbar icon, shown in Figure 3-46.

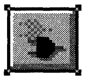


Figure 3-46 Move Toolbar Icon

   Selecting the Edit → Move command changes the pointer to a crosshair and enters Move mode, which is reflected in the PROcapture command line.

4. While holding the left mouse button down, drag the selected component to its new location.
5. Release the left mouse button to place the selected objects.

Moving Text

To move text, follow these steps:

1. Using the left mouse button, select the text.
2. Position the mouse inside the bounding box.
3. While holding the left mouse button down, drag the text to its new location.

Deleting Objects

You can remove an object from the schematic and store it in the clipboard so it can be pasted, or you can remove the item completely.

Removing Objects to the Clipboard

To delete an object from the schematic and remove it to the clipboard, select the **`Edit`** → **`Cut`** command.

Keyboard Shortcut: You can execute the Edit → Cut command by typing **`bcu`**↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the Edit → Cut command by clicking on the Cut toolbar icon, shown in Figure 3-47.

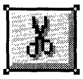


Figure 3-47 Cut Toolbar Icon

Removing Objects

To delete an object from the schematic without removing it to the clipboard, select the **`Edit`** → **`Clear`** command, or press the Delete key.

Keyboard Shortcut: You can delete an object by typing **`del`**↵ on the PROcapture command line.

Saving Schematics

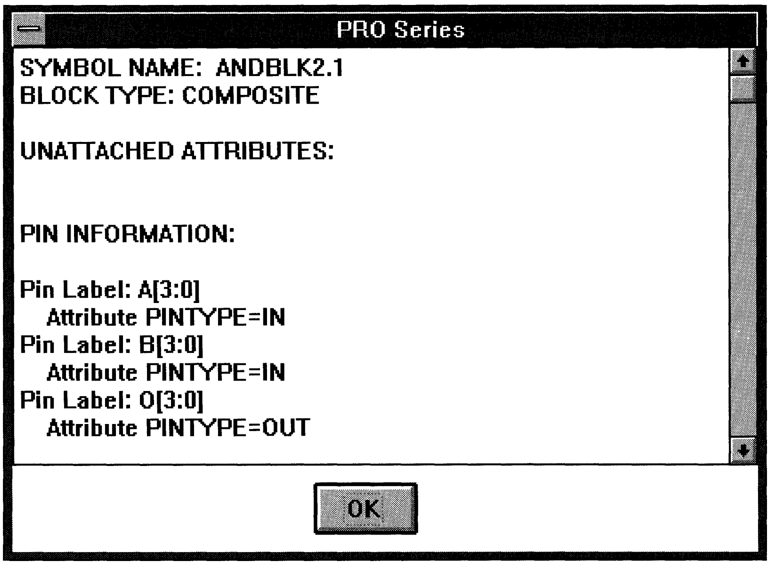
To save the schematic, select the **`File`** → **`Save`** command.

Keyboard Shortcut: You can execute the File → Save command by typing **`wri`**↵ on the PROcapture command line.

This command checks the schematic for any errors and then saves the schematic. If the following message is not displayed in the status window and a dialog box appears, correct the schematic accordingly and re-save when finished:

```
0 error(s) and 0 warning(s) in project
primary:schematic_name.1
```

This command generates a report, which is displayed in a PRO Series information dialog box. Figure 3-48 shows this report.



```
SYMBOL NAME:  ANDBLK2.1
BLOCK TYPE: COMPOSITE

UNATTACHED ATTRIBUTES:


PIN INFORMATION:

Pin Label: A[3:0]
   Attribute PINTYPE=IN
Pin Label: B[3:0]
   Attribute PINTYPE=IN
Pin Label: O[3:0]
   Attribute PINTYPE=OUT
```

Figure 3-48 PRO Series Report in Information Dialog Box

1. To view the entire report, use the down arrow at the bottom of the scroll bar.
2. Click on **OK**.

To save the schematic to another file, click on **`File`** → **`Save As`**.

Keyboard Shortcut: You can execute the File → Save As command by typing **`writet`**↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the File → Save As command by clicking on the Save As toolbar icon, shown in Figure 3-49.

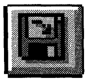


Figure 3-49 Save As Toolbar Icon

PROcapture prompts you to specify the block name and the sheet number. If the file already exists, PROcapture prompts you to verify that you want to overwrite it.

Copying Schematics

Rather than create a second schematic by adding components and drawing nets, you can copy the original schematic into the clipboard and then paste it into the new schematic window.

1. Open and name a new schematic.

   The new schematic window opens on top of the original schematic window.

2. Select the **`Window`** → **`Tile`** command.

   This command displays the open windows side by side in the workspace, as Figure 3-50 demonstrates.

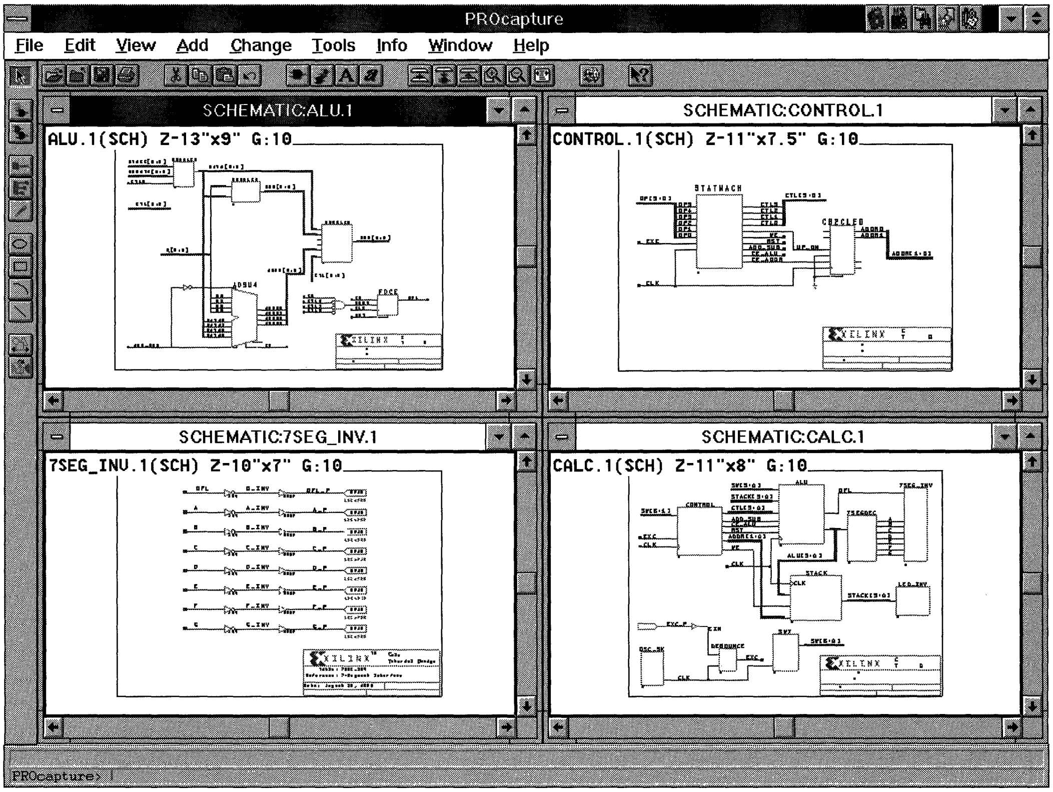

PROcapture
File Edit View Add Change Tools Info Window Help
SCHEMATIC:ALU.1
ALU.1(SCH) Z-13"x9" G:10
SCHEMATIC:CONTROL.1
CONTROL.1(SCH) Z-11"x7.5" G:10
SCHEMATIC:7SEG_INV.1
7SEG_INV.1(SCH) Z-10"x7" G:10
SCHEMATIC:CALC.1
CALC.1(SCH) Z-11"x8" G:10
PROcapture>

Figure 3-50 Tiled Windows

3. Click on the title bar of the original schematic window to set it as the current window.
4. Point the mouse above and to the left of all the components, nets, and buses in the original schematic.
5. Holding down the left mouse button, drag the mouse below and to the right of all the components, nets, and buses in the original schematic.
6. Release the left mouse button to select all components, nets, and buses in the area of the drag.
7. Select the **`Edit`** → **`Copy`** command.

8. Click on the title bar of the second schematic window to set it as the current window.
9. Select the **`Edit`** → **`Paste`** command.
10. Point the mouse to position the bounding box.
11. Click the left mouse button to place the components, nets, and buses.
12. To inspect the two schematics, select each and zoom in to the area where the components have been placed.

Navigating Through Schematics

PROcapture offers a number of options that allow you to view different aspects of a schematic.

- To view a schematic block through the hierarchy, select the block with the left mouse button, and click on **`View`** → **`Push Into Schematic`**. Because it pushes down only one level, you may need to select this command repeatedly to descend through several levels of hierarchy.

  Keyboard Shortcut: You can execute the View → Push Into Schematic command by typing **`psc`**↵ on the PROcapture command line once you select the block to be pushed into.

  Toolbar Shortcut: You can execute the View → Push Into Schematic command by clicking on the Push Into Schematic toolbar icon, shown in Figure 3-51.

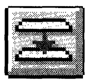


Figure 3-51 Push Into Schematic Toolbar Icon

  The View → Push Into Schematic command pushes the symbol's schematic onto the top of the active window's display stack and displays the schematic. The view of the block is magnified and centered on the screen.

- To view a symbol block through the hierarchy, select the symbol with the left mouse button, and click on **`View`** → **`Push Into`**

Symbol. The view of the symbol is now magnified and centered on the screen.

Keyboard Shortcut: You can execute the View → Push Into Symbol command by typing **psy**↵ on the PROcapture command line once you select the block to be pushed into.

Toolbar Shortcut: You can execute the View → Push Into Symbol command by clicking on the Push Into Symbol toolbar icon, shown in Figure 3-52.

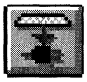


Figure 3-52 Push Into Symbol Toolbar Icon

- To return to the level of hierarchy present in the active window before the last Push command was issued, click on **View → Pop**.

  Keyboard Shortcut: You can execute the View → Pop command by typing **po**↵ on the PROcapture command line.

  Toolbar Shortcut: You can execute the View → Pop command by clicking on the Pop Schematic toolbar icon, shown in Figure 3-53.

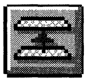


Figure 3-53 Pop Schematic Toolbar Icon

- To view the next sheet of a schematic, select **View → Push Next Sheet**.

  Keyboard Shortcut: You can execute the View → Push Next Sheet command by typing **psh**↵ on the PROcapture command line.

Printing Schematics

You can print the schematic or symbol in the active window by clicking on **File → Print**.

Toolbar Shortcut: You can execute the File → Print command by clicking on the Print toolbar icon, shown in Figure 3-59.

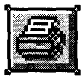


Figure 3-54 Print Toolbar Icon

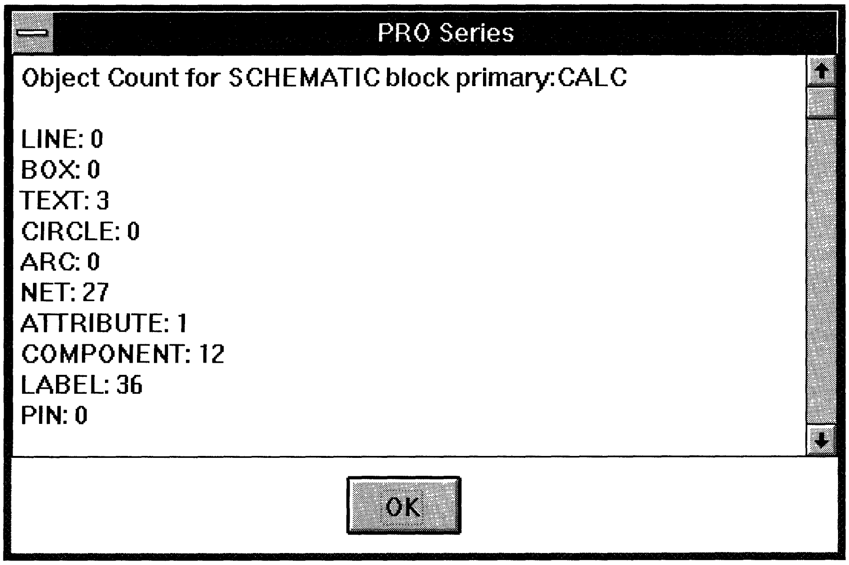
Obtaining Information

PROcapture offers four commands that allow you to obtain information about the objects in your design.

- To obtain a count of all the object types and the number of segments, joints, and connections in the active schematic or symbol window, click on `Info → Object Count`.

  Keyboard Shortcut: You can execute the Info → Object Count command by typing **oc**↵ on the PROcapture command line.

  A window similar to that in Figure 3-55 appears.



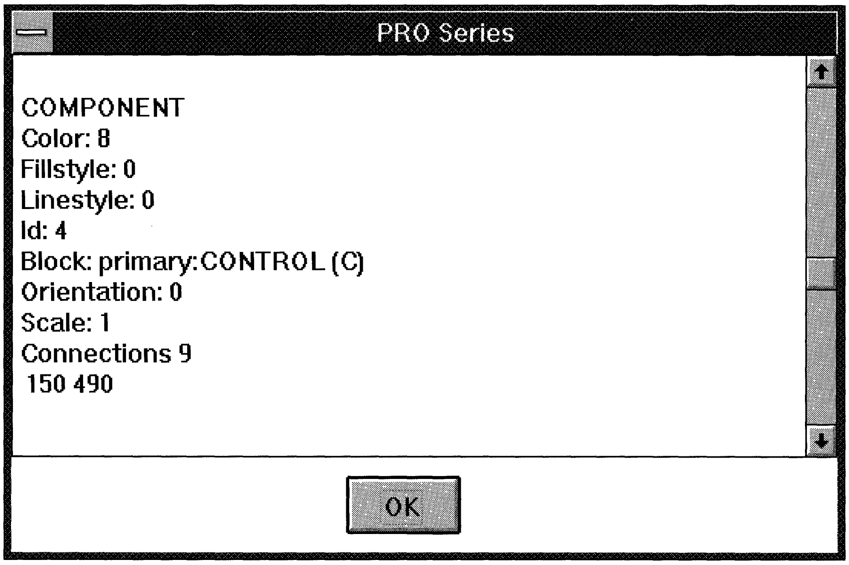
```
Object Count for SCHEMATIC block primary:CALC

LINE: 0
BOX: 0
TEXT: 3
CIRCLE: 0
ARC: 0
NET: 27
ATTRIBUTE: 1
COMPONENT: 12
LABEL: 36
PIN: 0
```

Figure 3-55 Object Count Window

- To obtain detailed information about a particular object, such as the component shown in Figure 3-56, select the object with the left mouse button and click **`Info`** → **`Object Detail`**.

  Keyboard Shortcut: You can execute the Info → Object Detail command by typing **`od`**↵ on the PROcapture command line.

  Figure 3-56 gives an example of the Object Detail window.



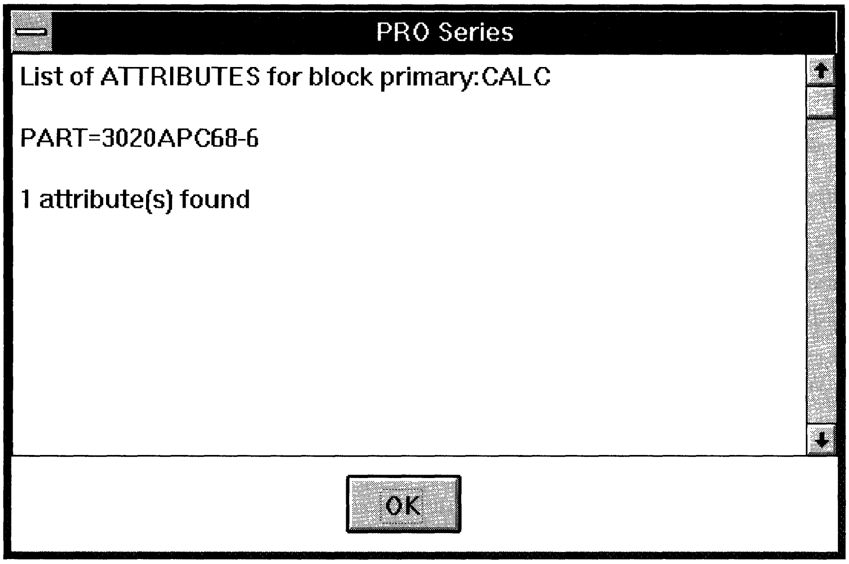
```
COMPONENT
Color: 8
Fillstyle: 0
Linestyle: 0
Id: 4
Block: primary:CONTROL (C)
Orientation: 0
Scale: 1
Connections 9
 150 490
```

Figure 3-56 Object Detail Window

- To list the attributes for the current block, click on **`Info`** → **`Object Attributes`**. You can also click on an object to display the attributes attached to that object.

  Keyboard Shortcut: You can execute the Info → Object Attributes command by typing **`al`**↵ on the PROcapture command line.

  A list of attributes appears, as shown in Figure 3-57.



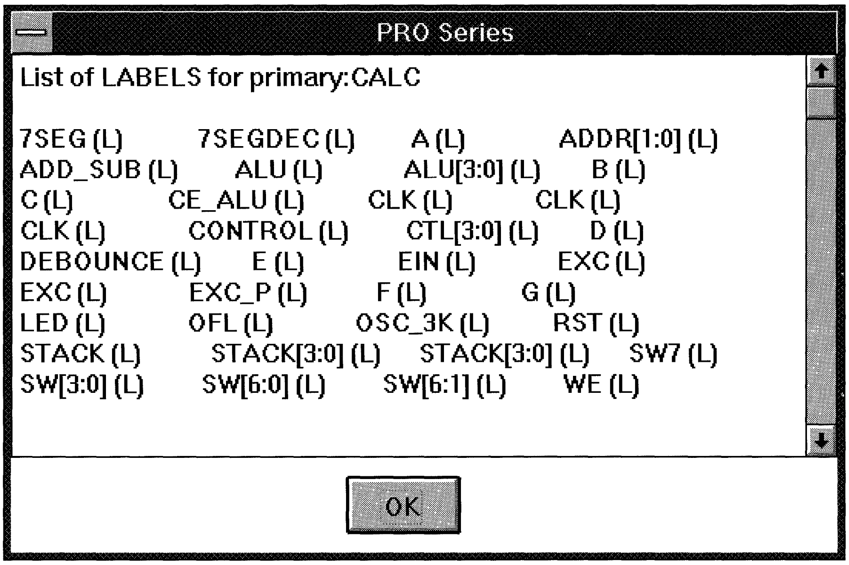
```
List of ATTRIBUTES for block primary:CALC

PART=3020APC68-6

1 attribute(s) found
```

Figure 3-57 Object Attributes Window

- To list the labels for the current block, click on **`Info`** → **`Object Labels`**. You can also click on an object to display the labels attached to that object.

  Keyboard Shortcut: You can execute the Info → Object Labels command by typing **`ll`**↵ on the PROcapture command line.

  A window resembling that in Figure 3-58 displays a list of labels.



List of LABELS for primary:CALC

7SEG (L)	7SEGDEC (L)	A (L)	ADDR[1:0] (L)
ADD_SUB (L)	ALU (L)	ALU[3:0] (L)	B (L)
C (L)	CE_ALU (L)	CLK (L)	CLK (L)
CLK (L)	CONTROL (L)	CTL[3:0] (L)	D (L)
DEBOUNCE (L)	E (L)	EIN (L)	EXC (L)
EXC (L)	EXC_P (L)	F (L)	G (L)
LED (L)	OFL (L)	OSC_3K (L)	RST (L)
STACK (L)	STACK[3:0] (L)	STACK[3:0] (L)	SW7 (L)
SW[3:0] (L)	SW[6:0] (L)	SW[6:1] (L)	WE (L)

Figure 3-58 Object Labels Window

Closing Schematics

To close a design, click on **`File`** → **`Close`**.

Keyboard Shortcut: You can execute the File → Close command by typing **`wcl`**↵ on the PROcapture command line.

Toolbar Shortcut: You can execute the File → Close command by clicking on the Close toolbar icon, shown in Figure 3-59.

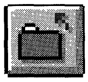


Figure 3-59 Close Toolbar Icon

Converting a Design

The Unified Libraries provide a means of effortless migration between Xilinx architectures. Because the shared components in the various libraries have the same names and symbol definitions, all you have to do to convert a design from one family to another is to run XAltran. XAltran changes the current library alias of each primitive used in the schematics of the current project to the desired target library alias.

As an example, you may be designing a circuit that targets an XC3000 part, so each primitive in your schematic references the XC3000 library using the (XC3000) alias. Later on in the design cycle, you decide that you would like to take advantage of the features offered by the XC4000 family of devices. Instead of modifying the schematic manually to target this new family, you can use XAltran to convert the primitive aliases from (XC3000) to (XC4000). Now all the primitives reference the XC4000 library.

To convert a design in the Powerview environment, use the Altran program. Once you use Altran to convert the design schematics, you must edit the viewdraw.ini file, adding the target library to the library list at the bottom.

For PRO Series, use the following procedure to convert your design.

1. To convert all the schematics in the current project, click on the PROflow Design Entry icon, shown in Figure 3-60.

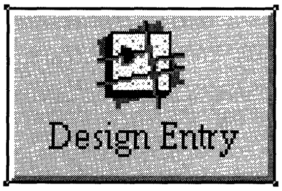


Figure 3-60 Design Entry Icon

The Design Entry dialog box shown in Figure 3-61 appears.

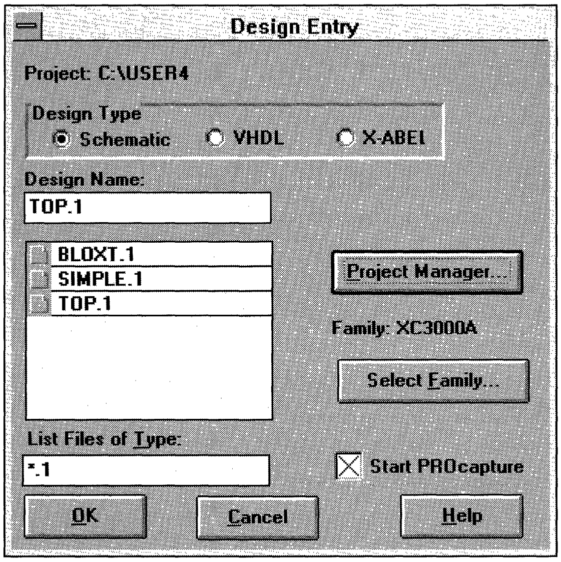


Figure 3-61 Design Entry Dialog Box

2. Click on the **`Select Family`** button.

   As indicated in Figure 3-62, the Select Family dialog box displays the current family setting.

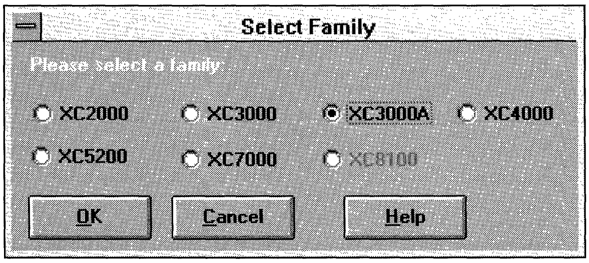
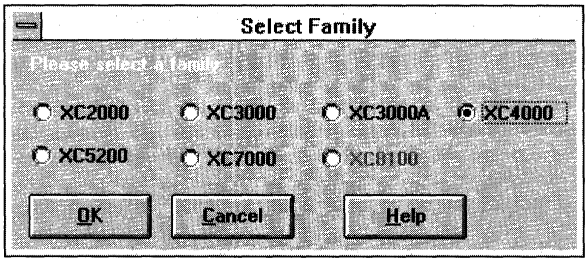


Figure 3-62 Select Family Dialog Box

3. Set the target family, as illustrated in Figure 3-63.

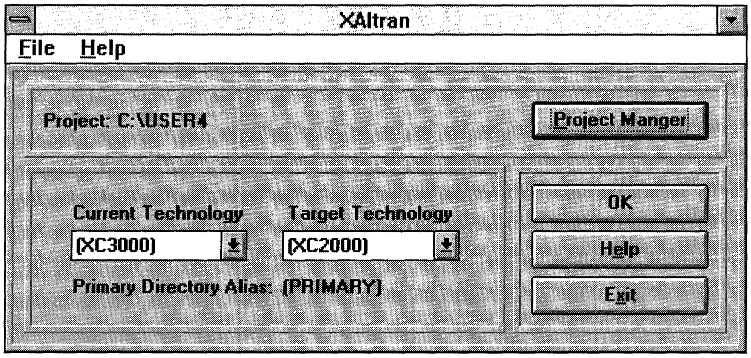


Selected family: [✓ XC4000]

Figure 3-63 Selecting the Target Family

4. Click on **OK**.

   The Select Family dialog box closes, and the XAltran dialog box appears, as shown in Figure 3-64.



Project: C:\USER4

Current Technology (XC3000)

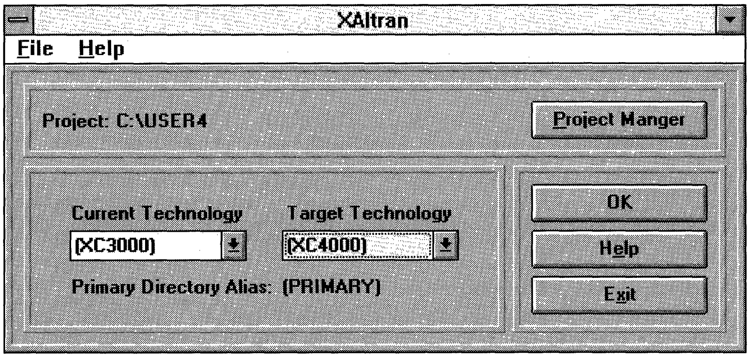
Target Technology (XC2000)

Primary Directory Alias: (PRIMARY)

Figure 3-64 XAltran Dialog Box

The XAltran dialog box is divided into three panels. The first panel displays the current project and enables you to invoke the PRO Series Project Manager so you can change your project. The second panel contains the controls that allow you to select the current and target technologies. Set the current technology to the family used in the schematics found in the current project. Set the target technology to the family to which you want to convert the schematics.

5. Using the left mouse button, open the **Target Technology** pull-down list box to display the target technologies from which to choose.
6. While holding down the left mouse button, point the mouse at the desired family.
7. Release the left mouse button to choose the target technology, as Figure 3-65 demonstrates.



Project: C:\USER4

Current Technology: [XC3000]

Target Technology: [XC4000]

Primary Directory Alias: [PRIMARY]

Figure 3-65 Selecting the Target Technology

The Target Technology pull-down list box is updated with the selected family.

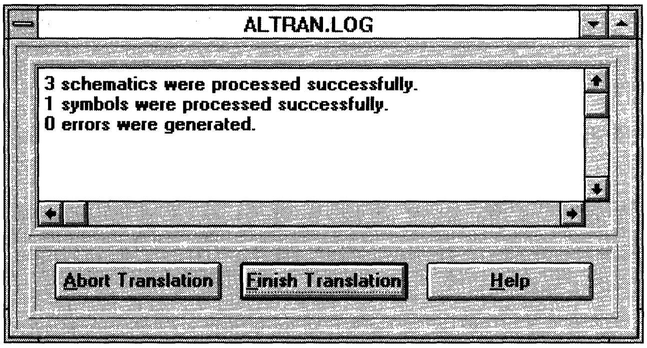
Note: If "No Alias" appears as the Primary Directory Alias at the bottom of the second panel, XAltran asks you if you would like to add a primary alias to the current project. You must add a primary alias to proceed. Adding this alias places the (primary) alias on all user-created symbols in the primary library being translated.

8. To convert the schematics in the current project, click on **OK**.

   XAltran closes, and an MS DOS box opens, which displays the command being executed as well as the standard output from the Altran program.

9. Once the conversion process completes, press any key to close the MS DOS box.

   The Altran log file is displayed in the window shown in Figure 3-66.



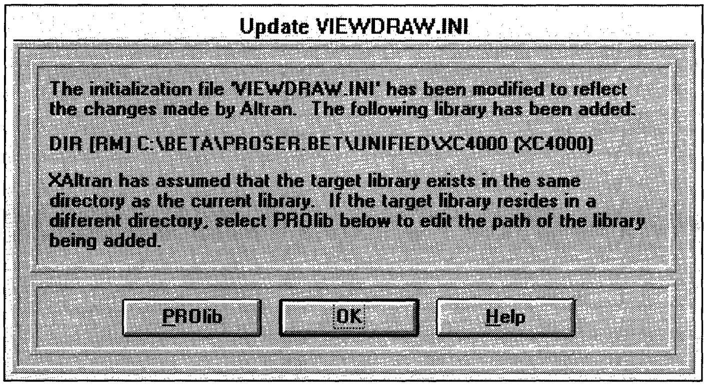
```
3 schematics were processed successfully.
1 symbols were processed successfully.
0 errors were generated.
```

Figure 3-66 Altran Log Screen

10. If the Altran conversion completed successfully, click on **`Finish Translation`** on the log screen.

Note: Should errors occur during the conversion, select Abort Translation and follow the instructions on the screen. Correct any schematic errors before reprocessing.

Selecting the Finish Translation command closes the Altran log file and displays the Update Viewdraw.ini dialog box, as shown in Figure 3-67.



The initialization file 'VIEWDRAW.INI' has been modified to reflect the changes made by Altran. The following library has been added:

DIR [RM] C:\BETA\PROSER.BET\UNIFIED\XC4000 (XC4000)

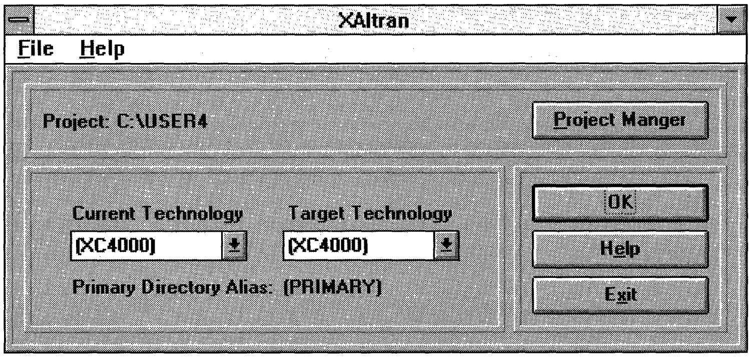
XAltran has assumed that the target library exists in the same directory as the current library. If the target library resides in a different directory, select PROlib below to edit the path of the library being added.

Figure 3-67 Update Viewdraw.ini Dialog Box

The Update Viewdraw.ini dialog box displays the new library that accesses the primitives to be used by the converted schematics. XAltran assumes that the target library resides in the same directory as the current library. If the target library is not located in the assumed directory, click on PROlib to modify the target library manually.

11. If the assumed library is correct, click on **OK**.

    The Update Viewdraw.ini dialog box closes, and the XAltran dialog box is updated, as Figure 3-68 illustrates.



XAltran

File Help

Project: C:\USER4

Current Technology: (XC4000)

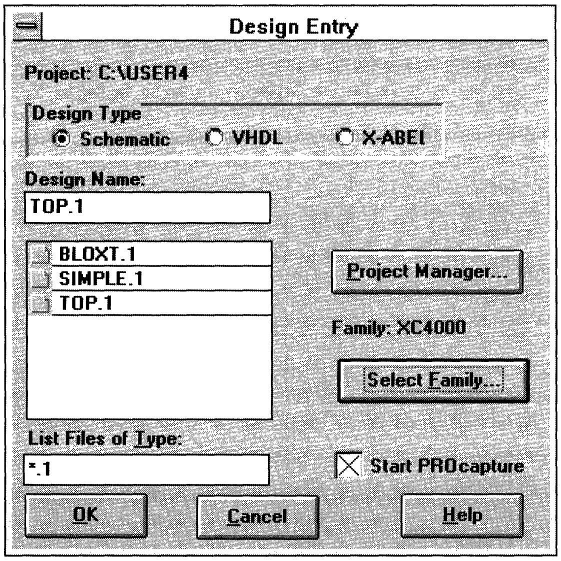
Target Technology: (XC4000)

Primary Directory Alias: (PRIMARY)

Figure 3-68 Updated XAltran Dialog Box

12. Click on `Exit` to close XAltran.

    Figure 3-69 displays the updated Design Entry dialog box.



Design Entry

Project: C:\USER4

Design Type
[✓ Schematic] VHDL X-ABEL

Design Name:
TOP.1

BLOXT.1
SIMPLE.1
TOP.1

Family: XC4000

List Files of Type:
*.1

[✓ Start PROcapture]

Figure 3-69 Updated Design Entry Dialog Box

13. To verify the conversion, select a schematic and click on **OK**.

Now PROflow invokes PROcapture and opens the desired schematic. Only components in the current technology that have equivalents in the target technology are translated. Those components that do not have an equivalent in the target technology do not appear in the converted schematics. For example, an XC4000 RAM component has no equivalent in an XC3000 device. Therefore, if you convert an XC4000 design with RAM components to an XC3000 design, the RAM components appear as white boxes on the converted schematic.

Viewlogic Interface Guide

Functional Simulation

Chapter 4

Functional Simulation

Functional simulation is an effective means of identifying logic errors in your design before it is implemented into a Xilinx device. Because the design has not yet been placed and routed, timing information for the design is not available, so the simulator tests the logic in the design using unit delays. You should simulate the functionality after your design has been entered to verify that the circuit logic is correct before mapping, placing, and routing take place. Finding errors before routing your design saves debugging time later in the design process.

This chapter describes how to use PROflow to create a simulation network and how to use PROsim to perform a functional simulation. It also describes how to use PROwave to view the simulation signals in a waveform format. The *Viewlogic Tutorials* manual illustrates the steps involved in functional simulation.

Simulation Procedure

It is recommended that you perform the initial simulation by issuing the simulation commands manually instead of using a command file. Then, once you have defined the sequence of commands, you can save the log of the session in a command file and use this file to re-simulate the design whenever you make a design change. The sequence of steps in this chapter reflects this methodology.

The typical procedure for performing a functional simulation is as follows:

1. Create the simulation network (VSM file).
2. Start PROsim.
3. Load the VSM file into PROsim.

4. Simulate the device's startup sequence.
5. Manually enter the simulation commands.
6. Optionally, run a command file.
7. Start PROwave.
8. View the waveforms produced by the simulation.
9. Repeat steps 5, 6, and 8 until the design is verified.

The rest of the chapter discusses these steps in detail.

Creating the Simulation Network

The first step in the functional simulation process is to create the simulation network (VSM file), which is loaded into PROsim to simulate the design. If the design being processed contains X-BLOX, RAM, ROM, or Xilinx ABEL modules, PROflow runs the XSimMake program to create the VSM file. For straight schematic designs, PROflow runs the VSM program instead to create it. PROflow runs XSimMake for all XC7000 EPLD designs.

1. If PROflow is not already running, double-click on the PROflow icon in the Program Manager XACT*step* program group.
2. Click on the Functional Simulation PROsim icon, shown in Figure 4-1:

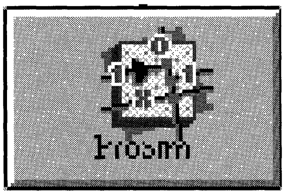
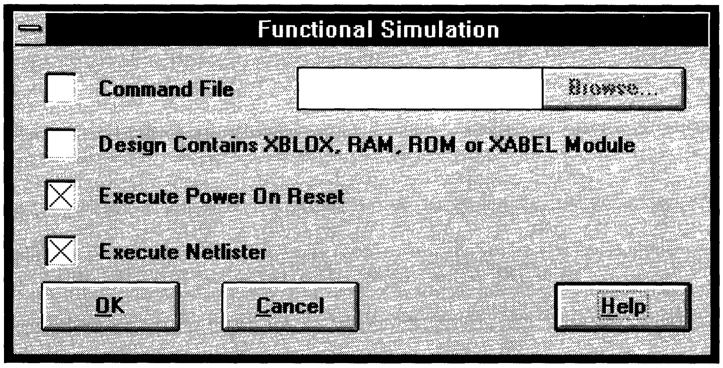


Figure 4-1 PROsim Icon

The Functional Simulation dialog box appears, pictured in Figure 4-2.


Functional Simulation

[ ] Command File [Browse...]

[ ] Design Contains XBLOX, RAM, ROM or XABEL Module

[✓] Execute Power On Reset

[✓] Execute Netlister

Figure 4-2 Functional Simulation Dialog Box

3. Set the following options where appropriate.
   - Command File instructs PROflow to run the simulation commands contained in a command file. Type in the name and the path name of this file, or click on **`Browse`** to locate it. By default, a command file is not specified.
   - Design Contains XBLOX, RAM, ROM, or XABEL Module instructs PROflow to run XSimMake to generate the VSM file for designs with RAM, ROM, X-BLOX, and Xilinx ABEL modules. If this option is not selected, PROflow uses the Viewlogic VSM program to create the simulation network by default.
   - Execute Power On Reset instructs PROflow to execute the commands necessary to simulate the target device's startup sequence. If this option is not selected, PROflow does not execute the commands necessary to simulate the startup sequence. Table 4-1 shows the global reset signals used for all architectures. Do not check this box if the command file contains the necessary commands to simulate startup or if your design does not contain any flip-flops. PROflow simulates startup by default.

Note: If the startup sequence is not simulated and the design contains flip-flops, the outputs of all flip-flops will always be X.

Table 4-1 Global Set/Reset Signals

Family	Global Set/ Reset Signal	Polarity
XC2000/L	GR	Active-Low
XC3000/A/L	GR	Active-Low
XC4000/A/D/H	GSR	Active-High
XC5200	GR	Active-High
XC7000	PRLD	Active-High

- Execute Netlister instructs PROflow to generate the simulation network. PROflow generates the simulation network by default.

  If you have not made any design changes and simply want PROsim to simulate using an existing simulation network, deselect the Execute Netlister option.

4. After setting the desired options, click on **OK**.

   If the Execute Netlister option is set, PROflow runs the necessary programs to create the simulation network. PROflow then invokes Notepad to allow you to view the netlister log file. Should the log file be too large for Notepad, use another text editor to review the file.

5. Verify that no errors or warnings were generated.

   If the netlister did not complete successfully, bring up PROcapture and correct the portion of the schematic that generated the error or warning message, then re-create the simulation network using the PROsim icon in PROflow as you did previously.

6. Select the **`File`** → **`Exit`** option to close Notepad.

Once you close Notepad, PROflow invokes PROsim on the simulation network.

After PROsim processes the net name equivalents, PROflow asserts the global reset net for 100 ns, then de-asserts it. This operation is required for simulation; it simulates the device's startup sequence when the device resets or sets its flip-flops. If these steps are not performed, the simulation output will not be correct; all flip-flop outputs remain in an indeterminate state, even though valid inputs and clocks are applied.

Finally, once the startup sequence has been simulated, PROflow executes the command file, if specified.

Creating Vectors

To aid in simulation, you can create vectors to group similar signals together. This step is particularly helpful when you want to group signals into buses other than those defined in your design. Also, you can create vectors to allow you to merge buses easily without changing the schematic. You can then assign these vectors values in PROsim and view them in PROwave.

At the `PROsim>` prompt, type **`vector`** *vector_name signal_list*↵ for each vector to be created, where the first signal in the list is the most significant bit (MSB), and the last signal in the list is the least significant bit (LSB).

PROsim confirms the Vector command.

Adding Signals, Buses, and Vectors to the Waveform

Now that the simulation netlist has been loaded into PROsim and you have defined the vectors, you can specify the signals, buses, and vectors to be displayed in PROwave.

To add signals to the waveform, follow these steps.

1. At the `PROsim>` prompt, type **`s_wave`**↵.

   PROsim prompts for the PROwave data stream file name, giving the default name of *design*.wfm in square brackets.

2. Press ↵ to accept the default name, or type a new name followed by ↵.

   By default, both functional and timing simulation create a waveform file called *design*.wfm. To avoid overwriting the functional simulation waveform file, specify *design*f.wfm for your functional simulation waveform and *design*t.wfm for your timing simulation waveform file.

3. At the `Node(s)/Vector(s)` prompt, type the list of desired signal, bus, or vector names, separated by spaces.

PROsim confirms the Wave command.

Note: Later, you will open PROwave, which will display the signals just added. If you specified a waveform file other than the default, you must use that name in PROwave to view the signals just added.

Defining Clocks, Standard Inputs, and Bidirectionals

After you define the signals to be viewed in PROwave, you must set up the design inputs and give them values to be simulated. There are three general types of stimuli: clocks, standard inputs, and bidirectionals.

Defining a Clock

You can use the Clock and Step commands to generate most periodic waveforms. For more "irregular" waveforms, see the description of the Every command in the Viewlogic documentation.

To define a clock as a periodic waveform, follow these steps.

1. At the `PROsim>` prompt, type **`clock`** *clock_name pattern*↵.

   Pattern is the sequence of highs (1s) and lows (0s) that defines a signal's waveform for one period.

   Here is an example of the Clock command:

   ```
   clock clk2 1 1 1 1 0 1
   ```

2. At the `PROsim>` prompt, type **`step`**↵ to accept the default step size of 100 ns, or type **`step`** *timevalue***`ns`**↵ to change the step size. For example, to set a step size of 10 ns, type the following:

   ```
   step 10ns
   ```

   The step size (*timevalue*) is the length of one step of the clock definition. For example, if you defined a new clock by typing the following, it would have a 50% duty cycle with a period of 400 ns:

   ```
   clock newclk 0 0 1 1↵
   ```

Defining Standard Input Values

Now that you have defined the clock as a periodic waveform, you can set the signals to their initial values.

1. To assign a value to a signal, type `{`**`h`**`|`**`l`**`}` *signal_name*↵ at the `PROsim>` prompt, where H is High and L is Low.
2. To assign a value to a vector or bus, use the Assign command at the `PROsim>` prompt. The syntax is the following:

   `assign` *vector_name value*

   Vector_name is the name of the bus or vector, and *value* is the combined set of values assigned to each signal that composes the bus or vector. Here is an example:

   ```
   assign sw 1111111
   ```

Defining Bidirectional Input Values

Bidirectionals are pins that can be forced to supply internal circuitry a value or that are released during simulation to allow the internal circuitry to output a value.

1. To assign a bidirectional a value, use the same commands given in the "Defining Standard Input Values" section.
2. To allow the internal circuitry to drive the signal, type **`release`** *signal_name*, where *signal_name* is the name of the signal, bus, or vector to be released.

Note: If a signal is released and nothing internal drives it, the value of the signal will be X.

Simulating the Design Inputs

Once you have defined the design inputs, the next step is to simulate or apply those values throughout the design.

Type **`sim`** *timevalue units*↵.

If the units are not specified, PROsim assumes the default scalar of 0.1 ns. For example, `sim 100` is equivalent to 10 ns.

If you now view the schematic in PROcapture, all the signals in the design display known values. The values that PROcapture displays in boxes are those forced by the assignment commands entered in PROsim.

Re-Creating Previous Simulation

During simulation, PROsim creates a log file that contains a history of every command issued in that session, as well as the PROsim status of the commands. You can edit this log file and save it to a command file that you can then use to re-create the previous simulation.

1. Once you have finished simulating, close PROsim.
2. Invoke Notepad and open the viewsim.log file.
3. Save the file to a *filename*.cmd file.

Invoking PROwave

A good way to view a large number of signals simultaneously is to open a waveform window in PROwave.

1. Click on the PROwave icon in PROflow, shown in Figure 4-3.

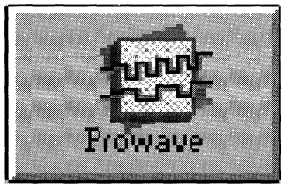
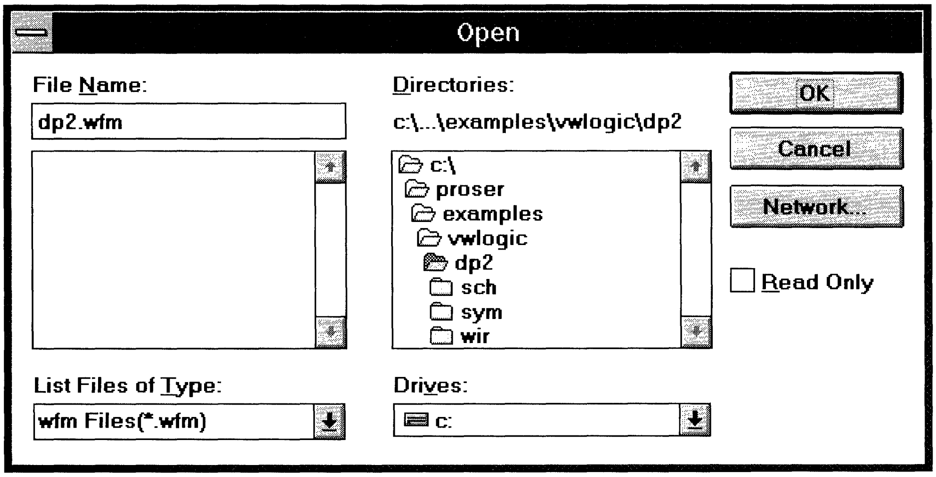


Figure 4-3 PROwave Icon

Selecting the PROwave icon invokes PROwave and displays the Open dialog box, shown in Figure 4-4.


Open

File Name:
dp2.wfm

Directories:
c:\...\examples\vwlogic\dp2

c:\
proser
examples
vwlogic
dp2
sch
sym
wir

List Files of Type:
wfm Files(*.wfm)

Drives:
c:

Figure 4-4 Open Dialog Box

2. In the File Name list box, select the waveform display file.

   The name of the waveform file is the name you used when you added the signals to the waveform. The default name is *design*.wfm.

Note: Be sure to select the file that you specified when you added the signals in the "Adding Signals, Buses, and Vectors to the Waveform" section, or the link between PROsim and PROwave will not be established.

3. Click on **OK**.

   The Open dialog box closes, and the specified waveform display file opens.

Note: If you do not select a WFM file from the File Name list box but instead enter a new name, a stimulus file is created. A stimulus file is used to create a stimulus command file and does not annotate values during simulation.

Changing the Display Radix in PROwave

The default PROwave display radix for grouped signals is hexadecimal. You may want to change the display radix to a binary, octal, or decimal base.

1. Point the mouse at the group text in the list of signals on the left side of the waveform window, and select it by clicking the left mouse button.
2. Select the **`Waveform`** → **`Set Radix`** → **`Decimal Radix`**, **`Octal Radix`**, or **`Binary Radix`** command to change the vector display to an alternative base. You can also change the display value by first selecting the signal and then selecting the corresponding toolbar icon. The binary radix icon is shown in Figure 4-5.

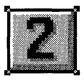


Figure 4-5 Binary Radix Toolbar Icon

The toolbar icons for the other radices are shown in the "PROwave Commands" chapter.

Changing the Display Radix in PROcapture

You can also change the radix of grouped signals displayed in PROcapture by setting the radix in PROsim so that it is written to the waveform display file.

To set the radix of a bus, type **`radix`** *radix bus_name*↵ at the `PROsim>` prompt.

Viewlogic Interface Guide

Design Implementation

Printed in U.S.A.

Chapter 5

Design Implementation

Once you complete functional simulation, you are ready to implement your design. Implementation is performed by the Design Manager, which you invoke from PROflow. The Design Manager first translates the design into a flattened Xilinx Netlist Format (XNF) file, then interfaces to the Flow Engine to implement the design. The Flow Engine optimizes, places, and routes the design, creates timing simulation data, and creates physical design data for downloading. This chapter describes how to use the Design Manager and the Flow Engine to translate and implement your design.

Within its own project, the Design Manager manages implementation versions and revisions. Each time that you change a schematic, it creates a new implementation version, and each time that you place and route the design, it creates a new revision of the current version to keep track of the various implementations. For more information on the Design Manager and how it manages versions and revisions, see the *Design Manager/Flow Engine Reference/User Guide.*

Invoking the Design Manager

You can invoke the Design Manager from PROflow or from the Program Manager XACT*step* program group.

1. Click on the Xilinx Implementation icon in PROflow, shown in Figure 5-1, or double-click on the Design Manager icon, shown in Figure 5-2, in the Program Manager XACT*step* program group.

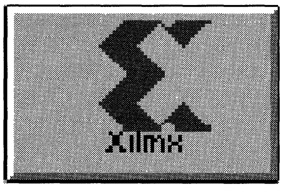


Figure 5-1 Xilinx Implementation Icon

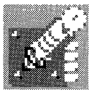


Figure 5-2 Design Manager Icon

The Design Manager window appears, as shown in Figure 5-3.

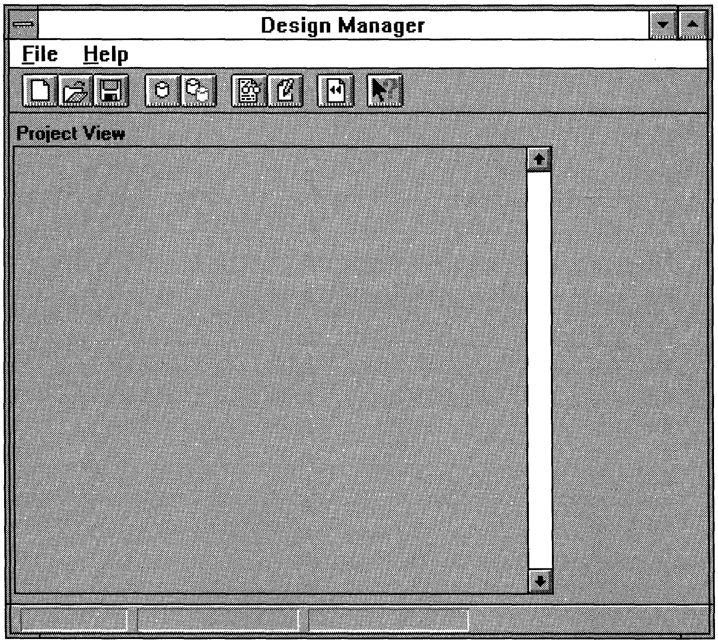


Figure 5-3 Design Manager Window

On the top of the window is the title bar, which identifies the tool as the Design Manager. Beneath it is the menu bar on which the pull-down menus appear. The full set of available menus does not appear until you load an implementation project into the Design Manager. The toolbar, which is located below the menu bar, displays icons that perform the same functions as the most commonly used menu commands. The Project View section of the window contains a graphical representation of the versions and revisions of the design. Finally, the status bar at the bottom of the window displays the family, part number, version number, and revision number of the current version.

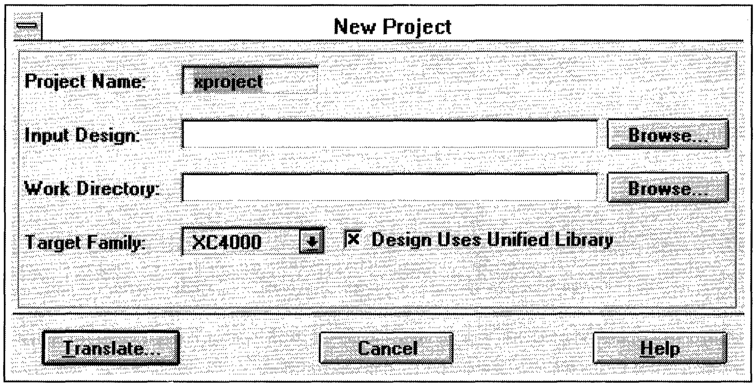
Creating the Xilinx Project and Its Initial Translation

Before you can begin implementing a design, you must create its Xilinx project, which is a different process from creating the design's Viewlogic project. The Xilinx project directories contain version and revision information for multiple runs of your design through the Xilinx implementation tools.

Follow these steps to create the Xilinx project.

1. Select the **`File`** → **`New Project`** command.

   The New Project dialog box appears, as shown in Figure 5-4.


New Project

Project Name: xproject

Input Design:

Work Directory:

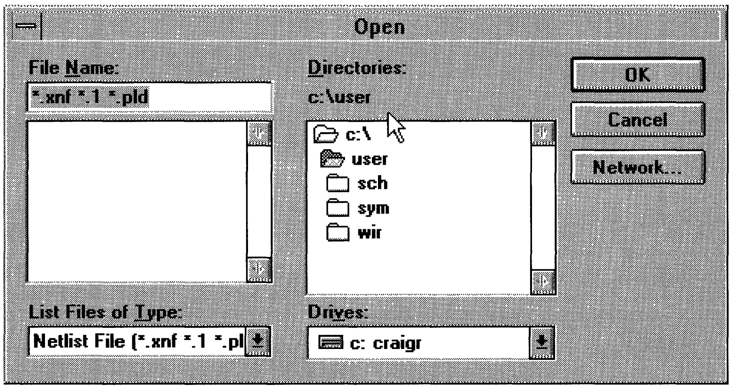
Target Family: XC4000

[✓ Design Uses Unified Library]

Figure 5-4 New Project Dialog Box

2. Specify the top-level design name in the Input Design field, or click on the **Browse** button.

   The Open dialog box appears, as shown in Figure 5-5, which allows you to select XNF files, schematic files, and PLD files.



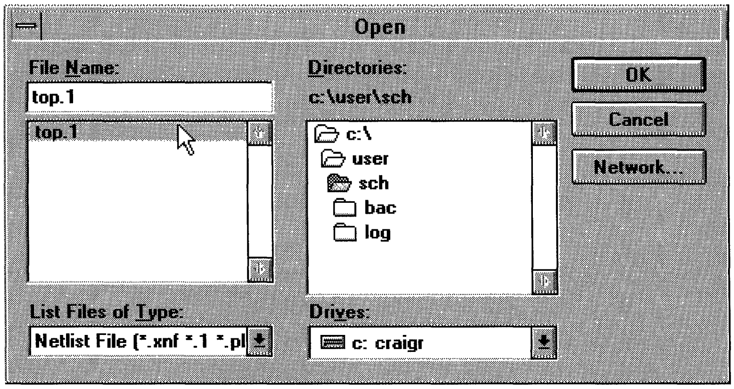
File Name: *.xnf *.1 *.pld

List Files of Type: Netlist File (*.xnf *.1 *.pl

Figure 5-5 Open Dialog Box

3. Select the input file.

   Figure 5-6 shows the selection of a top-level schematic file called top.1.



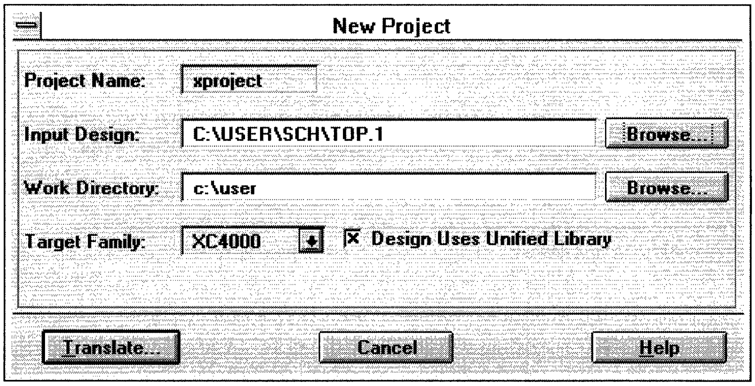
File Name: top.1

List Files of Type: Netlist File (*.xnf *.1 *.pl

Figure 5-6 Schematic File Selected in Open Dialog Box

4. Click on **OK**.

   The Open dialog box now closes, and the New Project dialog box is updated with the selected file, as demonstrated in Figure 5-7.



New Project

Project Name: xproject

Input Design: C:\USER\SCH\TOP.1

Work Directory: c:\user

Target Family: XC4000

[✓ Design Uses Unified Library]

Figure 5-7 Updated New Project Dialog Box

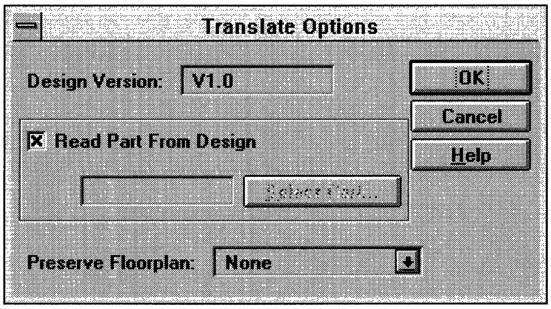
   The Work Directory field now displays the default value of *project*. The project directory, by default xproject, is placed in the work directory and contains the files created by the Design Manager when it translates designs and creates revisions.

5. In the Target Family field, select the family in which the device will be implemented.
6. If your design was not created with elements from the Unified Libraries, deselect the Design Uses Unified Library box.

   For more information on this dialog box, see the *Design Manager/ Flow Engine Reference/User Guide.*

7. Click on **`Translate`**.

   The Translate Options dialog box appears, as shown in Figure 5-8, to enable you to enter the version name, specify the design's part type, and select floorplan data to be carried forward, if any.



Design Version: V1.0

[✓ Read Part From Design]

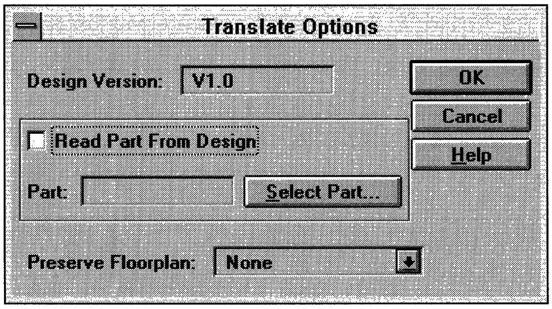
Preserve Floorplan: None

Figure 5-8 Translate Options Dialog Box

The Design Version field displays the version number of the design. For the first translation, V1.0 appears by default. With subsequent translations of the same design, the version number automatically increments.

8. If the input design does not specify a part, deselect the Read Part from Design check box.

   The Select Part button is now activated, as shown in Figure 5-9.



Design Version: V1.0

Read Part From Design

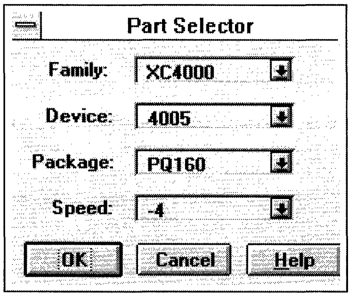
Part:

Preserve Floorplan: None

Figure 5-9 Activated Part Fields on Translate Options Dialog Box

9. Click on the **`Select Part`** button, or type the part number in the Part field.

   Selecting Select Part displays the Part Selector dialog box, as indicated in Figure 5-10.


Part Selector

Family: XC4000

Device: 4005

Package: PQ160

Speed: -4

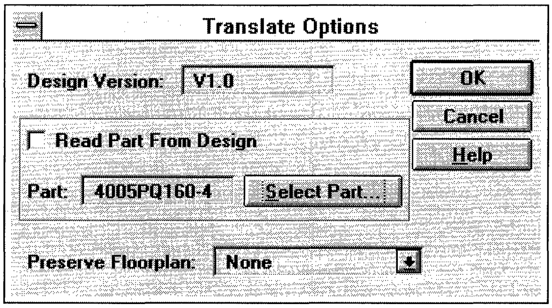
Figure 5-10 Part Selector Dialog Box

In this dialog box, the Device field reflects only the parts for the family selected in the Family field. Similarly, the Package field displays only those packages suitable for the device selected. The Speed grade field displays only those speed grades for the part and package selected.

10. Select the family, device, package, and speed grade appropriate for your device.

11. Click on **OK**.

    The Translate Options dialog box is now updated, as shown in Figure 5-11.


Translate Options

Design Version: V1.0

☐ Read Part From Design

Part: 4005PQ160-4

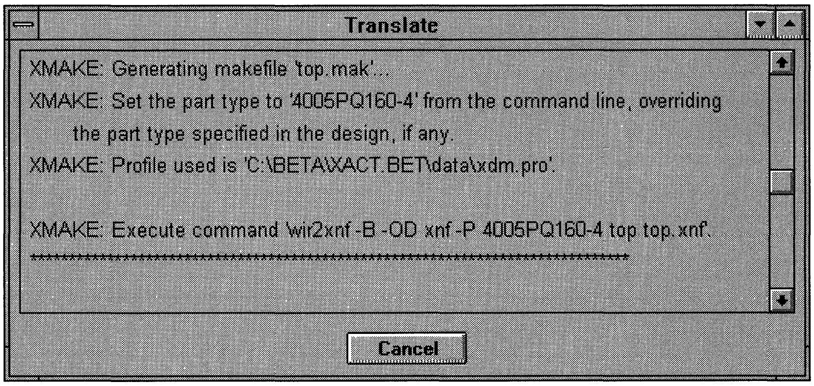
Preserve Floorplan: None

Figure 5-11 Updated Translate Options Dialog Box

12. To translate the design, select **OK**.

    The Translate status window opens, as shown in Figure 5-12.

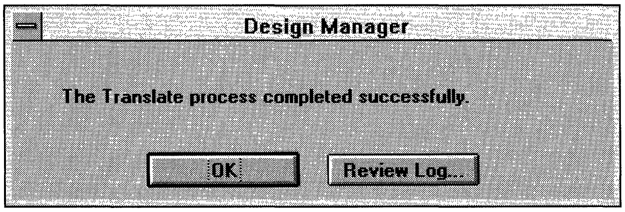
    This window displays the processing of the design so that you can monitor the progress of the translation.



```
XMAKE: Generating makefile 'top.mak'...
XMAKE: Set the part type to '4005PQ160-4' from the command line, overriding
    the part type specified in the design, if any.
XMAKE: Profile used is 'C:\BETA\XACT.BET\data\xdm.pro'.

XMAKE: Execute command 'wir2xnf -B -OD xnf -P 4005PQ160-4 top top.xnf'.
**********************************************************************
```

Figure 5-12 Translate Status Window

The Design Manager uses XMake for FPGAs and XEMake for EPLDs to translate the input design to a flattened file. They first translate the input design files to XNF files, then merge the XNF files into one flattened XNF file called *design*.xff. When they successfully complete the translation, the Design Manager displays the message box shown in Figure 5-13.

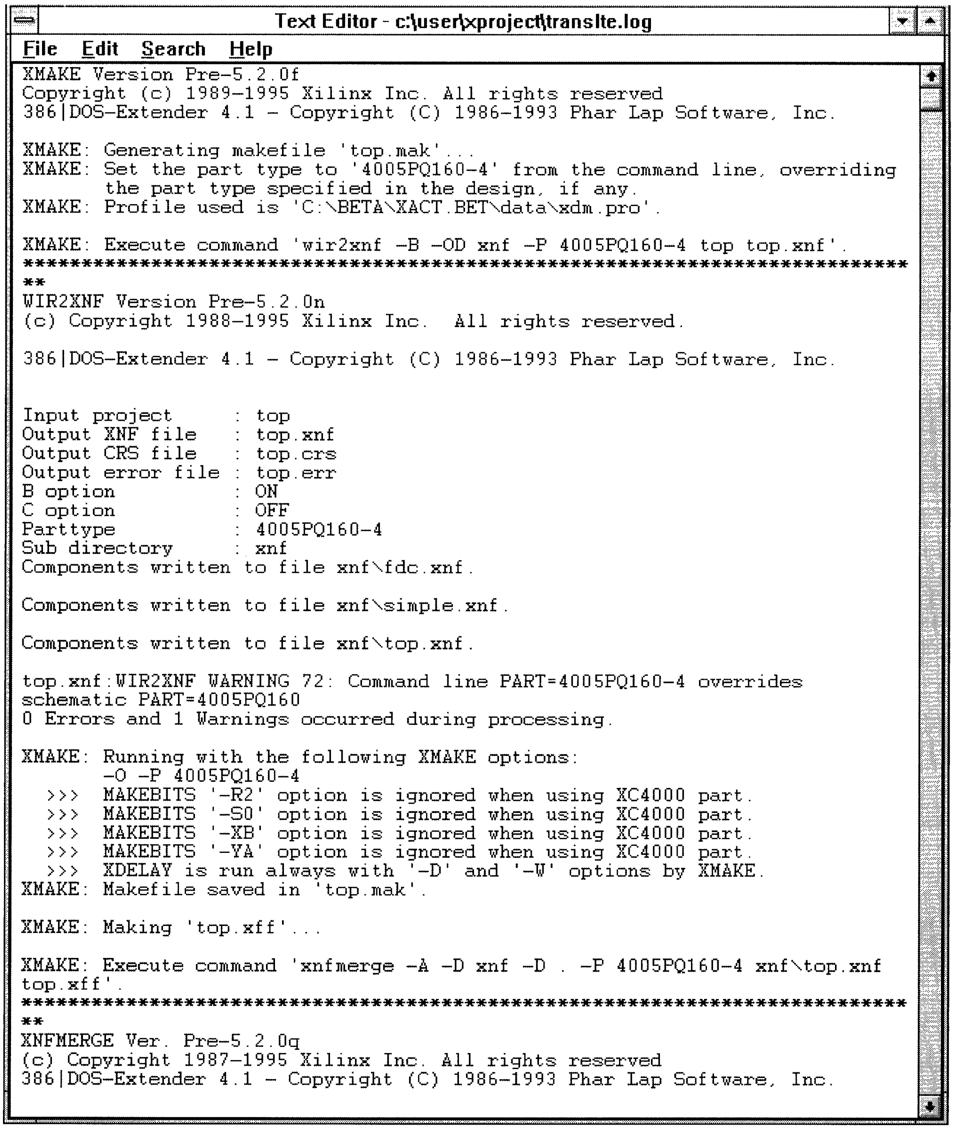


The Translate process completed successfully.

Figure 5-13 Design Manager Translation Message Box

13. To review the results of the translation, which are placed in a log file, click on **`Review Log`**.

The message box closes, and a text editor displays the log file, as shown in Figure 5-14.



```
XMAKE Version Pre-5.2.0f
Copyright (c) 1989-1995 Xilinx Inc. All rights reserved
386|DOS-Extender 4.1 - Copyright (C) 1986-1993 Phar Lap Software, Inc.

XMAKE: Generating makefile 'top.mak'...
XMAKE: Set the part type to '4005PQ160-4' from the command line, overriding
       the part type specified in the design, if any.
XMAKE: Profile used is 'C:\BETA\XACT.BET\data\xdm.pro'.

XMAKE: Execute command 'wir2xnf -B -OD xnf -P 4005PQ160-4 top top.xnf'.
****************************************************************************
**
WIR2XNF Version Pre-5.2.0n
(c) Copyright 1988-1995 Xilinx Inc.  All rights reserved.

386|DOS-Extender 4.1 - Copyright (C) 1986-1993 Phar Lap Software, Inc.


Input project      : top
Output XNF file    : top.xnf
Output CRS file    : top.crs
Output error file  : top.err
B option           : ON
C option           : OFF
Parttype           : 4005PQ160-4
Sub directory      : xnf
Components written to file xnf\fdc.xnf.

Components written to file xnf\simple.xnf.

Components written to file xnf\top.xnf.

top.xnf:WIR2XNF WARNING 72: Command line PART=4005PQ160-4 overrides
schematic PART=4005PQ160
0 Errors and 1 Warnings occurred during processing.

XMAKE: Running with the following XMAKE options:
       -O -P 4005PQ160-4
  >>>  MAKEBITS '-R2' option is ignored when using XC4000 part.
  >>>  MAKEBITS '-S0' option is ignored when using XC4000 part.
  >>>  MAKEBITS '-XB' option is ignored when using XC4000 part.
  >>>  MAKEBITS '-YA' option is ignored when using XC4000 part.
  >>>  XDELAY is run always with '-D' and '-W' options by XMAKE.
XMAKE: Makefile saved in 'top.mak'.

XMAKE: Making 'top.xff'...

XMAKE: Execute command 'xnfmerge -A -D xnf -D . -P 4005PQ160-4 xnf\top.xnf
top.xff'.
****************************************************************************
**
XNFMERGE Ver. Pre-5.2.0q
(c) Copyright 1987-1995 Xilinx Inc. All rights reserved
386|DOS-Extender 4.1 - Copyright (C) 1986-1993 Phar Lap Software, Inc.
```

Figure 5-14 Translation Results

14. To exit the text editor, select **`File`** → **`Exit`**.

    As Figure 5-15 indicates, the Design Manager is now updated to show the new project.

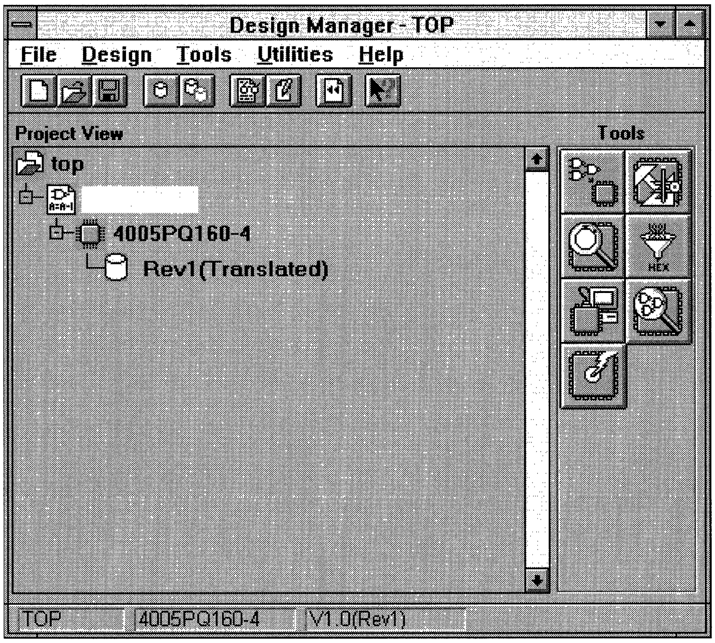


Figure 5-15 New Project in Design Manager Window

In the Project View section of the window, the new project contains a single version with an implementation revision. The status of the revision is "Translated." The Tools section of the window, which contains the icons of the Xilinx tools available for the new project, such as the Flooplanner and the Timing Analyzer, is now displayed. The menu bar is also updated to show all the available menus.

Implementing an FPGA Design

The Design Manager uses the Flow Engine to implement a design. There are several ways to invoke the Flow Engine; the following procedure demonstrates one method.

1. In the Design Manager window, use the left mouse button to select the revision to be implemented.

   The selected revision is highlighted.

2. To display the list of commands that can be performed on the revision, hold the right mouse down on the revision to be implemented, as shown in Figure 5-16.

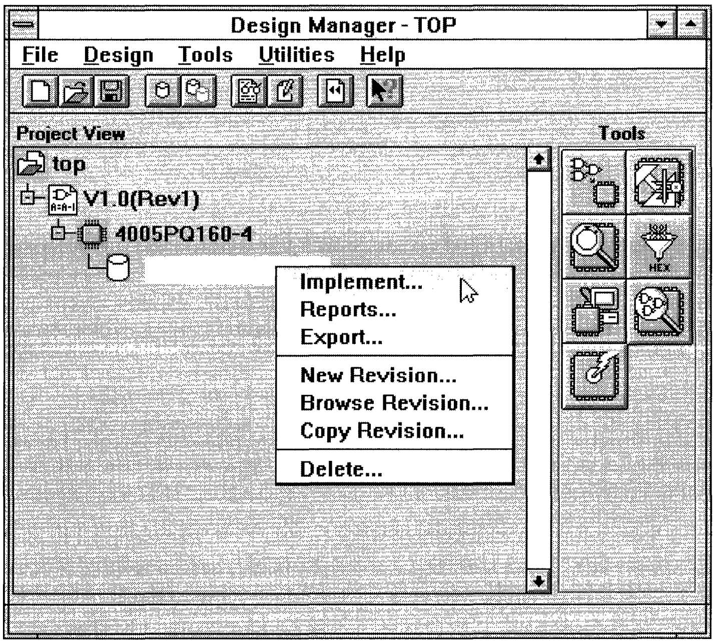
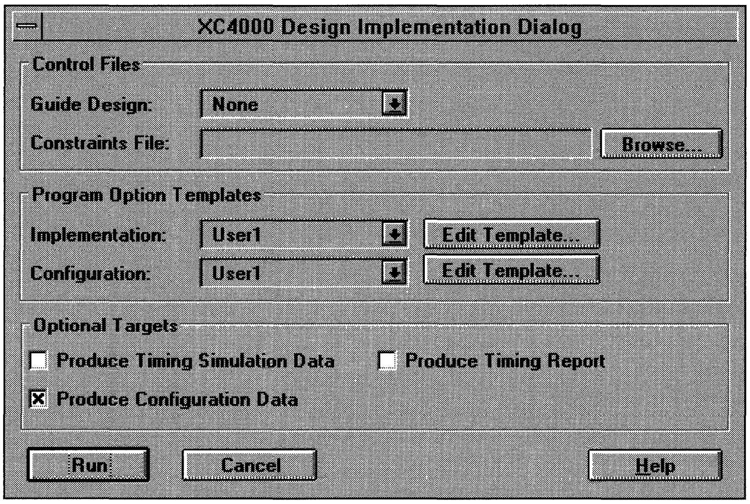


Figure 5-16 Revision Commands

3. Release the right mouse button on the `Implement` command.

   This command opens the Design Implementation dialog box, as shown in Figure 5-17.


XC4000 Design Implementation Dialog

Control Files

Guide Design: None

Constraints File:

Program Option Templates

Implementation: User1

Configuration: User1

Optional Targets

☐ Produce Timing Simulation Data

☐ Produce Timing Report

☒ Produce Configuration Data

Figure 5-17 Design Implementation Dialog Box

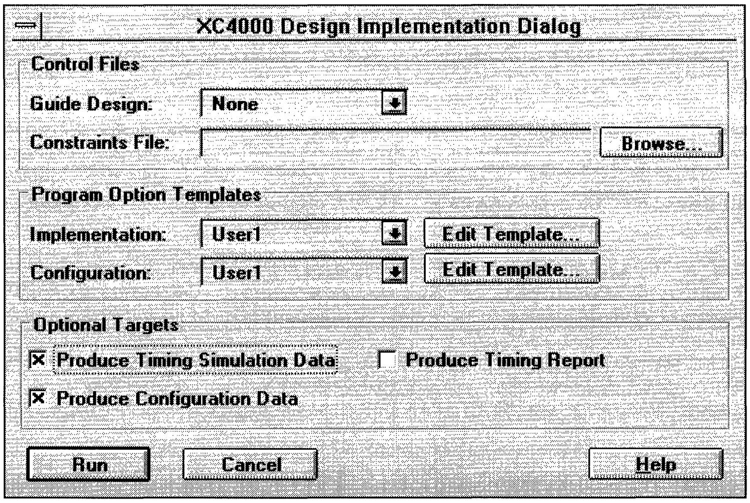
4. Select any control files necessary to translate the design.

   To choose a guide design file, click on the down arrow to the right of the Guide Design field. The valid guide designs in your project are listed.

   To use a constraints file, type the name and path in the Constraints File field, or use the **`Browse`** button.

5. Leave the Produce Configuration Data option turned on; this option generates the physical data needed to download the design.

6. To produce timing simulation data, click on the Produce Timing Simulation Data check box, as shown in Figure 5-18.


XC4000 Design Implementation Dialog

Control Files

Guide Design: None

Constraints File:

Program Option Templates

Implementation: User1

Configuration: User1

Optional Targets

[✓] Produce Timing Simulation Data

[ ] Produce Timing Report

[✓] Produce Configuration Data

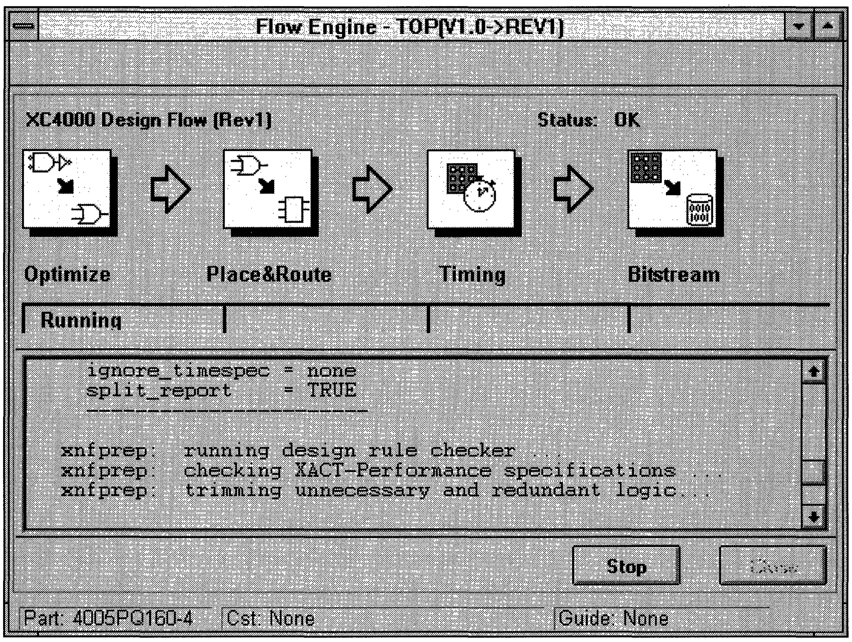
Figure 5-18 Selecting Produce Timing Simulation Data Check Box

The Design Manager creates configuration and timing simulation data in the Xilinx project area. Because PROflow does not have access to that area, the Design Manager exports the needed information to the Viewlogic project area. It exports this data each time that the design is implemented. When multiple implementations occur, the Viewlogic project area contains the data for the most recent implementation.

Note: For information on exporting previous implementation data, see the "Exporting Revision Data" section later in this chapter.

7. To implement the design, click on the **Run** button.

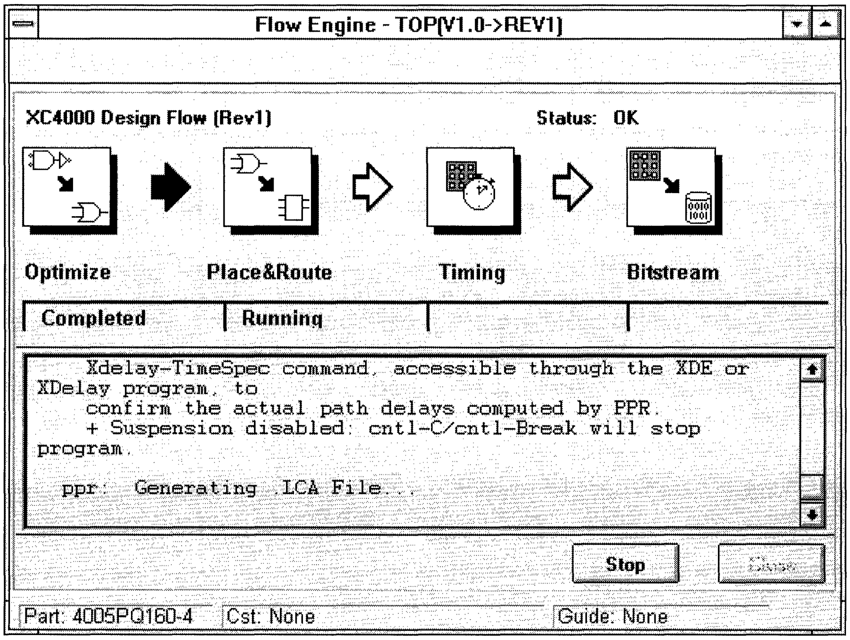
   This command closes the dialog box and opens the Flow Engine, which handles the processing of the design from optimization to the generation of the bitstream. As each step is completed, the Flow Engine window is updated with the processing status. The first step is the optimization step, which is performed by the XNFPrep program. The Flow Engine window shown in Figure 5-19 displays the optimization step.



```
     ignore_timespec = none
     split_report    = TRUE
     -----------------------

   xnfprep:  running design rule checker ...
   xnfprep:  checking XACT-Performance specifications ...
   xnfprep:  trimming unnecessary and redundant logic...
```

Figure 5-19 Flow Engine Window

Once optimization is completed, the Flow Engine places and routes the design, as reflected in the window in Figure 5-20. The place and route stage creates the layout and routes the interconnect.



```
    Xdelay-TimeSpec command, accessible through the XDE or
XDelay program, to
    confirm the actual path delays computed by PPR.
    + Suspension disabled: cntl-C/cntl-Break will stop
program.

  ppr: Generating LCA File...
```

Part: 4005PQ160-4 Cst: None Guide: None

Figure 5-20 Place and Route Phase of Implementation

Next, the Flow Engine submits the design for timing simulation and generates the bitstream. The completion of these processes produces the data necessary for PROflow to simulate timing and download the design.

8. To close the Flow Engine once processing is complete, click on **`Close`**.

   The Flow Engine closes, and the updated Design Manager is displayed, as shown in Figure 5-21.

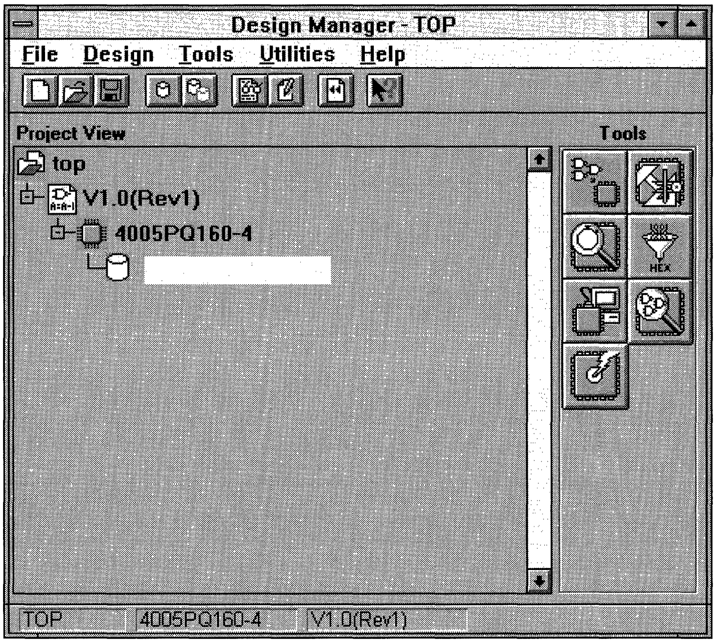


Figure 5-21 Configured Design Icon in Design Manager Window

Using the Report Browser

When you click on the Run button in the Design Implementation dialog box, the Report Browser is also displayed. During processing, the Report Browser is updated with reports as they are created. Figure 5-22 illustrates the Report Browser once processing by the Flow Engine is complete.

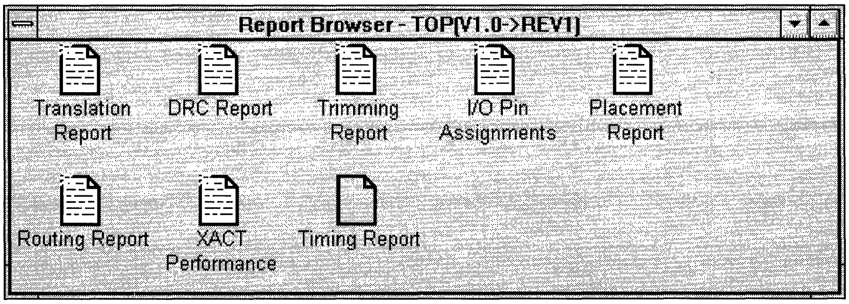


Figure 5-22 Report Browser (FPGA Reports)

The Report Browser provides access to the various reports generated when a design is processed. The Report Browser not only keeps track of which reports are available but also indicates if those reports are new. A star in the upper left corner of the report icon means that a report has not yet been read.

1. To review a report, double-click on its icon with the left mouse button.

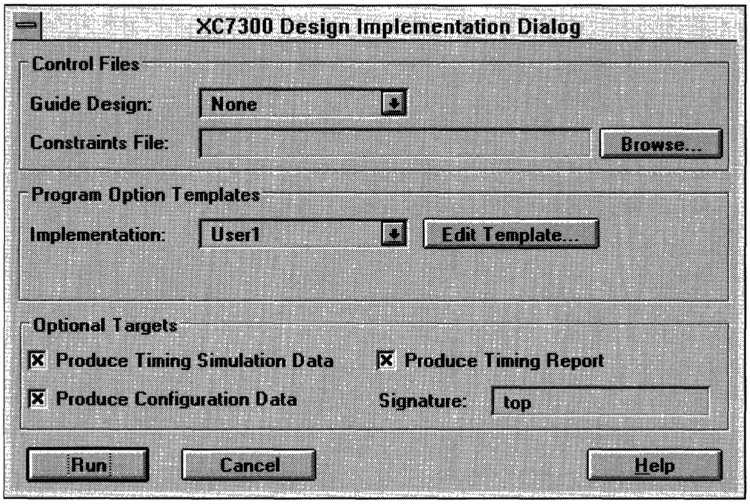
   The report opens in a text editor so you can review it. You can also edit the report and save it in a new area. Once you view a report, the star in the upper left corner disappears, signifying that the report has been reviewed.

2. To close the Report Browser, double-click on the Close box in the upper left-hand corner.

Implementing an EPLD Design

The Design Manager uses the Flow Engine to implement a design. There are several ways to invoke the Flow Engine; the following procedure demonstrates one method.

1. To implement an EPLD design, select **`Design`** → **`Implement`**. The XC7300 Design Implementation dialog box appears, as shown Figure 5-23.


XC7300 Design Implementation Dialog

Control Files

Guide Design: None

Constraints File:

Program Option Templates

Implementation: User1

Optional Targets

[X] Produce Timing Simulation Data
[X] Produce Timing Report
[X] Produce Configuration Data

Signature: top

Figure 5-23 XC7300 Design Implementation Dialog Box

2. Select any control files necessary to translate the design.

   To choose a guide design file, click on the down arrow to the right of the Guide Design field. The valid guide designs in your project are listed.

   To use a constraints file, type the name and path in the Constraints File field.

   By default, timing simulation data, configuration data, and a timing report are created during implementation.

3. Leave the Produce Timing Report, Produce Timing Simulation Data, and Produce Configuration Data options on.

4. Select the **Run** button in the Design Implementation dialog box.

   The Flow Engine and the Report Browser now display the stages in the design processing, as shown in Figure 5-24.

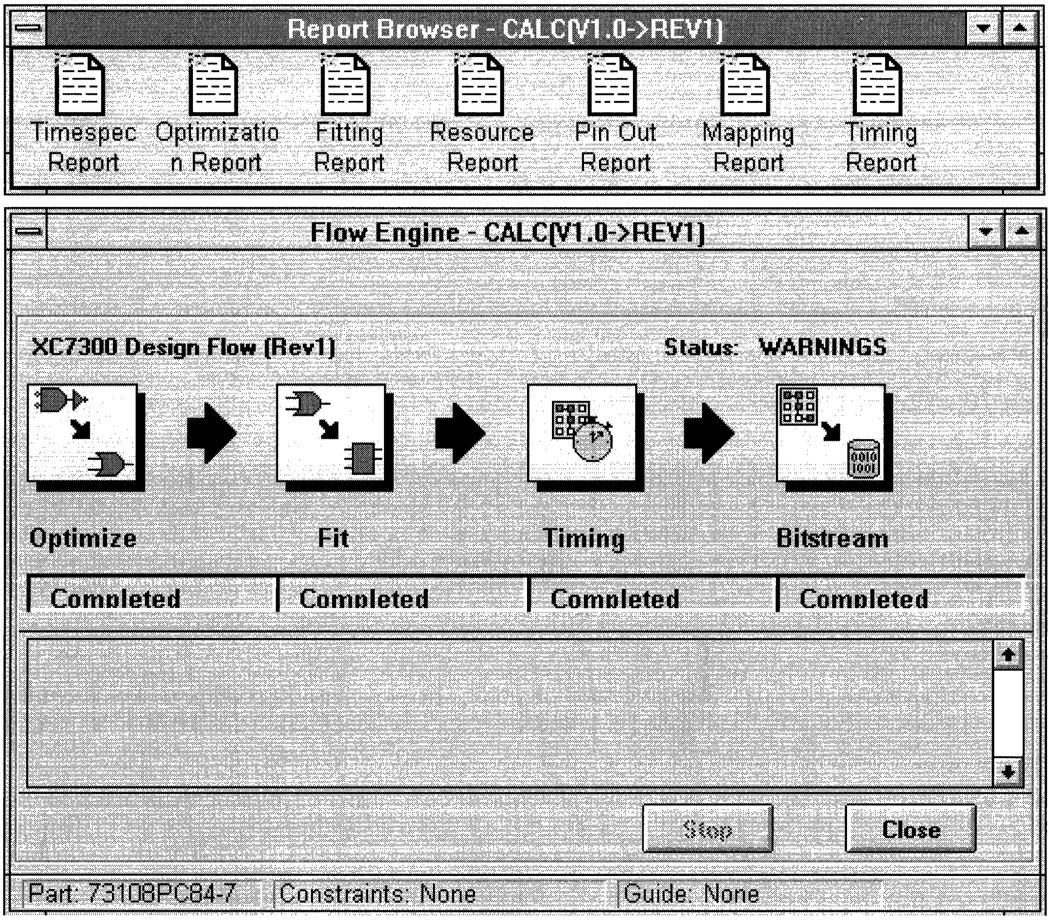


Figure 5-24 Flow Engine and Report Browser

The Design Manager creates configuration and timing simulation data in the Xilinx project area. Because PROflow does not have access to that area, the Design Manager exports the needed information to the Viewlogic project area. It exports this data each time that the design is implemented. When multiple implementations occur, the Viewlogic project area contains the data for the most recent implementation.

Note: On some machines with additional memory, running XEMake may take a long time because of memory management constraints; excessive processing time is not an indication that the machine has frozen.

5. When processing is complete, select the `Close` button.

You can now view the fitter reports, including the timing report, perform timing simulation, and program the device. Timing simulation is described in the "Timing Simulation" chapter. Device programming for EPLDs is described in the *HW130 Programmer User Guide.* For information on the Report Browser, see the "Using the Report Browser" section earlier in this chapter.

Exporting Revision Data

As you re-implement designs and create new revisions, the most recent revision data is placed in the Viewlogic project directory. PROflow can only access the most recent data, so if you want to use data from previous revisions, you must export it to the Viewlogic project, which is accessible to PROflow. Figure 5-25 displays a design version containing four revisions, two of which are configured. Because Rev2 was the most recent revision in which configuration data was produced, its data resides in the Viewlogic project directory. If you wanted to use the data in Rev1, you must have first exported it to the Viewlogic project.

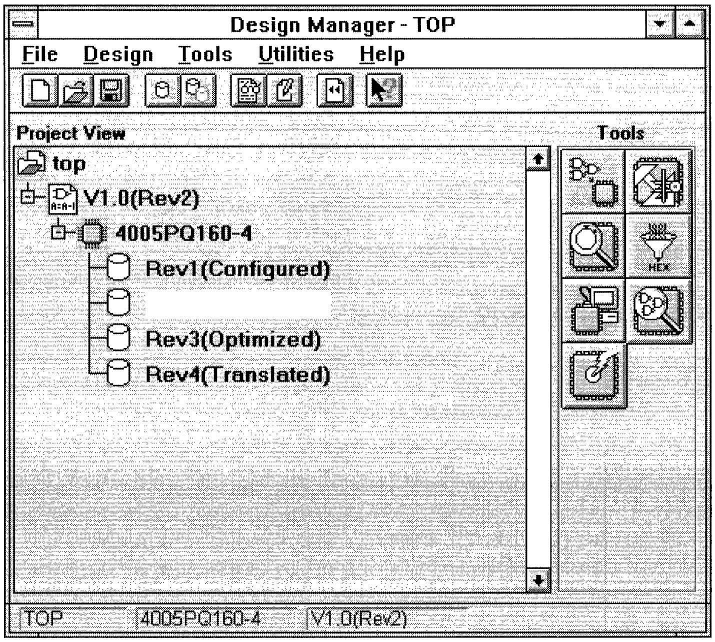


Figure 5-25 Four Revisions of a Version

1. With the left mouse button, select the revision to be exported.

   The selected revision is highlighted.

2. Hold the right mouse down on the revision to be exported to display the list of commands that can be performed on the revision, as shown in Figure 5-26.

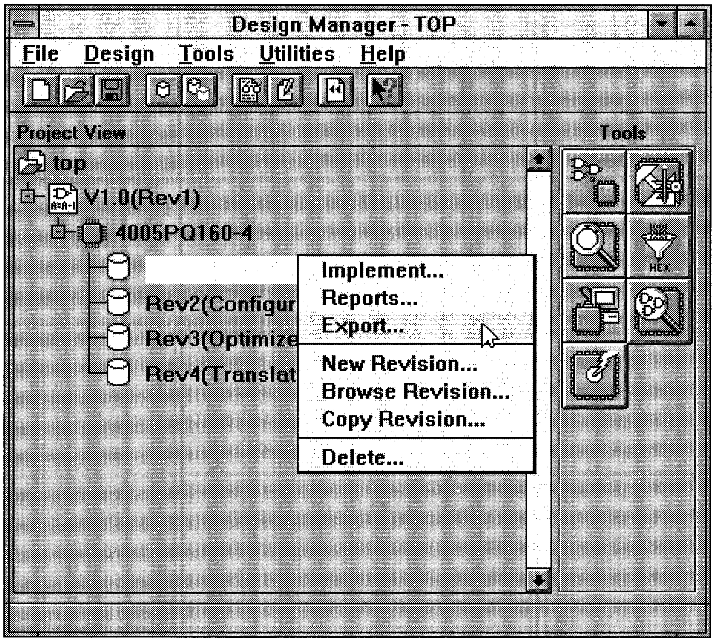
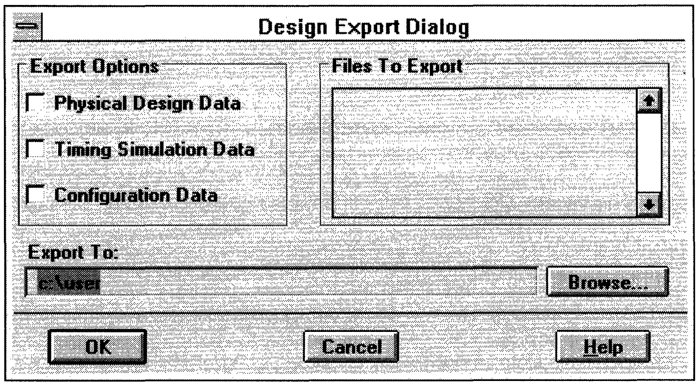


Figure 5-26 Revision Commands

3. Release the right mouse button on the **`Export`** command.

   This command opens the Design Export dialog box, as shown in Figure 5-27.


Design Export Dialog

Export Options
- Physical Design Data
- Timing Simulation Data
- Configuration Data

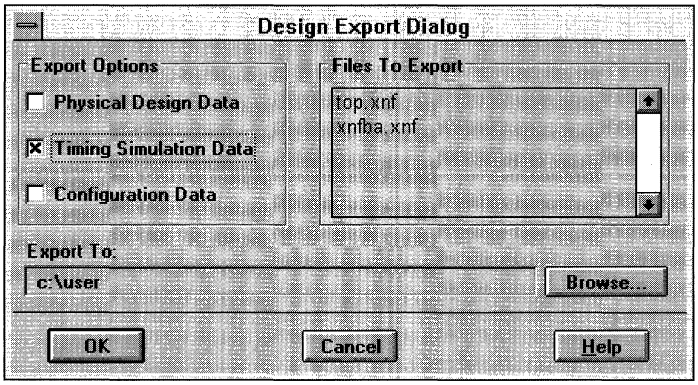
Files To Export

Export To: c:\user

Figure 5-27 Design Export Dialog Box

The Design Export dialog box allows you to select the types of data to place in the Viewlogic project directory. It includes physical design data, timing simulation data, and configuration data. When you select one or more of these types, the Files to Export list box is updated to show the actual names of the files being exported. In the Export To box, you can also select the destination directory, which by default is set to the Viewlogic project directory.

4. Click on the Timing Simulation Data check box, as shown in Figure 5-28.



Export Options: [ ] Physical Design Data, [✓ Timing Simulation Data], [ ] Configuration Data

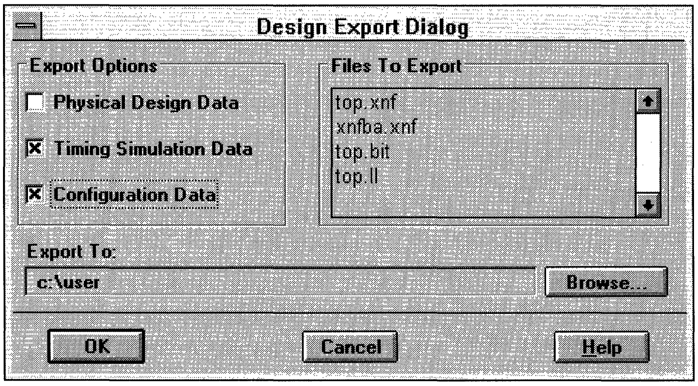
Files To Export:
top.xnf
xnfba.xnf

Export To: c:\user

Figure 5-28 Selecting Timing Simulation Data Option

When you select this option, the Design Manager places the *design*.xnf and xnfba.xnf files in the Viewlogic project directory. PROflow requires these files to create a timing simulation network.

5. Click on the Configuration Data check box, as shown in Figure 5-29.



Export Options: [ ] Physical Design Data, [✓ Timing Simulation Data], [✓ Configuration Data]

Files To Export:
top.xnf
xnfba.xnf
top.bit
top.ll

Export To: c:\user

Figure 5-29 Selecting Configuration Data Option

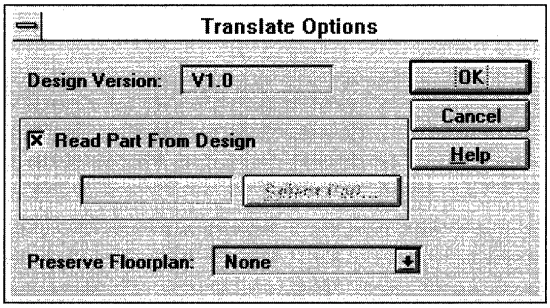
When you select this option, the Design Manager places the *design*.bit and *design*.ll files in the Viewlogic project directory. PROflow requires these files to download the design.

Translating the Design After Schematic Changes

Each time that you change the input design, you must create a new implementation version.

1. In the Design Manager window, click on **`Design`** → **`Translate`**.

   The Translate Options dialog box appears, as shown in Figure 5-30.



Design Version: V1.0

[✓] Read Part From Design

Preserve Floorplan: None

Figure 5-30 Translate Options Dialog Box

2. Follow the instructions given in the "Creating the Xilinx Project and Its Initial Translation" section, starting with step 8.
3. Re-implement the design by following the instructions in the "Implementing an FPGA Design" section for FPGAs or in the "Implementing an EPLD Design" section for EPLDs.

Viewlogic Interface Guide

Timing Simulation

Chapter 6

Timing Simulation

Timing simulation verifies a placed and routed design by using worst-case routing and block delay information. The delay information is extracted from the LCA or VMH file and passed to the back-annotated simulation netlist so it can be used during timing simulation. Timing simulation reduces the need for hardware debugging by determining whether or not the design works under worst-case conditions.

Timing simulation is also a useful tool for determining the device speed grade required for a particular application.

This chapter describes how to use PROflow to create a simulation network and how to use PROsim to perform a timing simulation. It also describes how use PROwave to view the simulation output signals in a waveform format. The *Viewlogic Tutorials* manual illustrates the steps involved in timing simulation.

Simulation Procedure

In most cases, you can use the command file generated during functional simulation to perform the timing simulation. For EPLDs, some of the internal nodes may not exist in the timing simulation netlist because of the optimization performed during the implementation process. You may need to make minor adjustments to the command file used to functionally simulate the design before it can be used to perform a timing simulation.

The typical procedure for performing a timing simulation is as follows.

1. Create the timing simulation network (VSM file).
2. Start PROsim.

3. Load the VSM file into PROsim.
4. Simulate the device's startup sequence.
5. Manually enter the simulation commands, which may be the same as those used during functional simulation, or execute the command file generated during functional simulation.
6. Optionally, run a new command file.
7. Start PROwave.
8. View the waveforms produced by the simulation.
9. Repeat steps 5, 6, and 8 as necessary to verify the timing information.

The rest of the chapter discusses these steps in detail.

Creating the Simulation Network

The first step in the timing simulation process is to create the simulation network (VSM file), which is loaded into PROsim to simulate the design.

1. After implementing the design using the Xilinx Implementation icon in PROflow, click on the Timing Simulation PROsim icon, shown in Figure 6-1. It appears below the Xilinx Implementation icon.

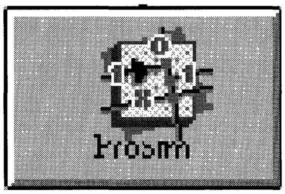
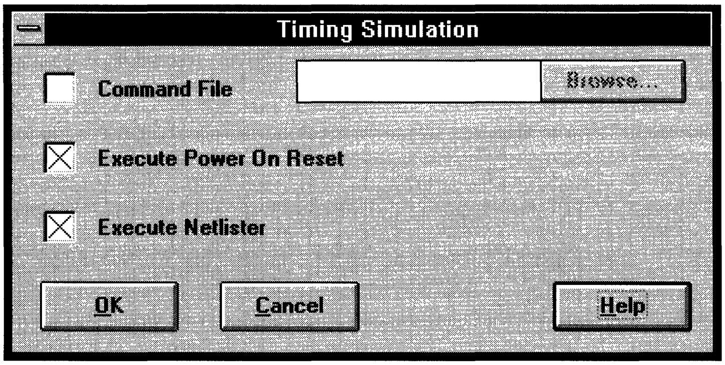


Figure 6-1 Timing Simulation PROsim Icon

The Timing Simulation dialog box appears, as shown in Figure 6-2.



[ ] Command File

[✓] Execute Power On Reset

[✓] Execute Netlister

Figure 6-2 Timing Simulation Dialog Box

2. Set the following options where appropriate.
   - Command File instructs PROflow to run the simulation commands contained in a command file. Type in the name and the path name of this file, or click on **`Browse`** to locate it. By default, a command file is not specified.
   - Execute Power On Reset instructs PROflow to execute the commands necessary to simulate the target device's startup sequence. If this option is not selected, PROflow does not execute the commands necessary to simulate the startup sequence. Table 6-1 shows the global reset signals used for all architectures. Do not check this box if the command file contains this option or if your design does not contain any flip-flops. PROflow simulates startup by default.

Note: If the startup sequence is not simulated and the design contains flip-flops, the outputs of all flip-flops will always be X.

Table 6-1 Global Set/Reset Signals

Family	Global Set/ Reset Signal	Polarity
XC2000/L	GR	Active-Low
XC3000/A/L	GR	Active-Low
XC4000/A/D/H	GSR	Active-High
XC5200	GR	Active-High
XC7000	PRLD	Active-High

- Execute Netlister instructs PROflow to generate the simulation network. PROflow first determines if the necessary files exist and then verifies that those files are for the current design translation. If PROflow detects timing data that is older than the current design translation, it displays a dialog box asking if you would like to proceed with the present data. To create new timing data, use the Xilinx Implementation icon.

  To create the timing simulation network, PROflow uses either XSimMake or RunTSim. The setting of the Design Contains XBLOX, RAM, ROM, or XABEL Module option in the Functional Simulation dialog box at the time of functional simulation determines which program PROflow uses. If you did not specify this option when you performed the functional simulation, PROflow runs RunTSim. If you specified this option, PROflow runs XSimMake. PROflow runs XSimMake for all XC7000 EPLD designs.

  If you have not re-implemented the design and simply want PROsim to simulate using an existing simulation network, deselect the Execute Netlister option.

3. To use a command file to perform a timing simulation, click on the Command File check box.

Note: If you do not have a command file available or simply wish to enter the simulation commands manually, see the following sections in the "Functional Simulation" chapter: "Creating Vectors," "Adding Signals, Buses, and Vectors to the Waveform," "Defining Clocks, Standard Inputs, and Bidirectionals," and "Simulating the Design Inputs."

The Command File field is now activated, as indicated in Figure 6-3, so that you can type in the name of the command file. The Browse button is also activated.

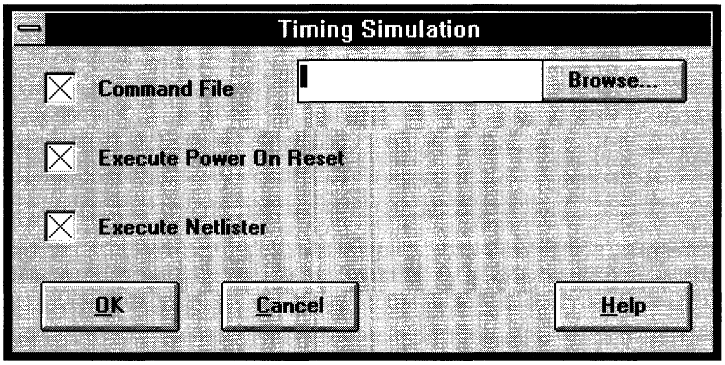


Figure 6-3 Selecting the Command File

4. Type in the name and path name of the desired command file, or click on **`Browse`**.

   The Command Files dialog box appears, shown in Figure 6-4.

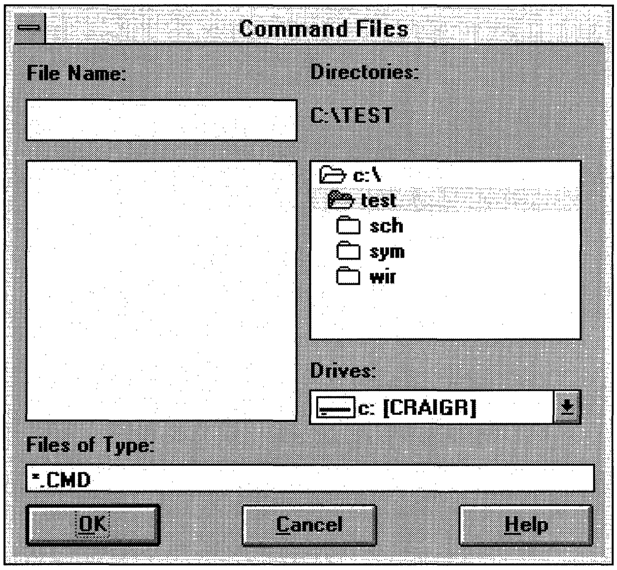


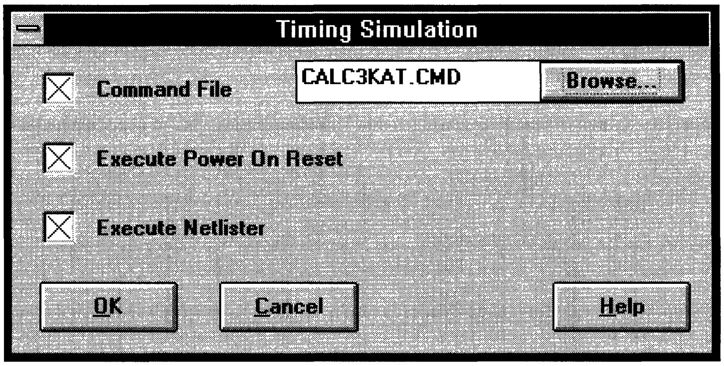
Figure 6-4 Command Files Dialog Box

5. Type in the name of the command file in the File Name field or select it from the list box. If you want to use the command file generated by the functional simulation, select the *filename*.cmd file.

Note: As noted in the "Functional Simulation" chapter, the default waveform file created during both functional and timing simulation is called *design*.wfm. To avoid overwriting the functional waveform file, specify a new name when adding signals to the timing waveform file. You can also do this by simply editing the functional command file, changing the waveform file name when the signals are added. You can then save the modified command file to a new file name so that both a functional and a timing command file can be used.

6. Click on **OK**.

   The Command Files dialog box closes, and the Timing Simulation dialog box is updated, as shown in Figure 6-5.



[✓ Command File] CALC3KAT.CMD

[✓ Execute Power On Reset]

[✓ Execute Netlister]

Figure 6-5 Updated Timing Simulation Dialog Box

Note: To view the PROsim commands, open the command file using Notepad or any text editor.

7. After setting the desired options, click on **`OK`**.

   The Timing Simulation dialog box closes, and PROflow prepares the timing simulation network. If the Execute Netlister option is set, PROflow runs the necessary programs to create the simulation network. PROflow then invokes Notepad to allow you to view the netlister log file.

   Should the log file be too large for Notepad, use another text editor to review the file.

8. Verify that no errors were issued during the processing.
9. Select the **`File`** → **`Exit`** option to close Notepad.

Once you close Notepad, PROflow invokes PROsim on the simulation network.

After PROsim processes the net name equivalents, PROflow asserts the global reset net for 100 ns, then de-asserts it. Finally, PROflow runs the specified command file. The command file sets up the vectors, initializes the inputs, and begins the simulation.

Once PROsim has finished processing the commands in the command file, you can continue entering commands manually, run another command file, or start PROwave to view the current waveforms.

Re-Creating the Previous Simulation

As with functional simulation, PROsim creates a log file during timing simulation that contains a history of every command issued in that session, as well as the PROsim status of the commands. You can edit this log file and save it to a command file that you can then use to re-create the previous timing simulation.

1. Once you have finished simulating, close PROsim.
2. Invoke Notepad and open the viewsim.log file.
3. Save the file to a *filename*.cmd file.

Invoking PROwave

To help verify the simulation, you can use PROwave to view the signal waveforms.

1. Click on the Timing Simulation PROwave icon in PROflow, shown in Figure 6-6.

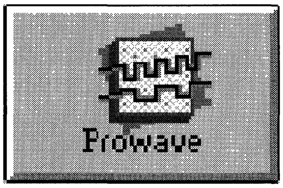
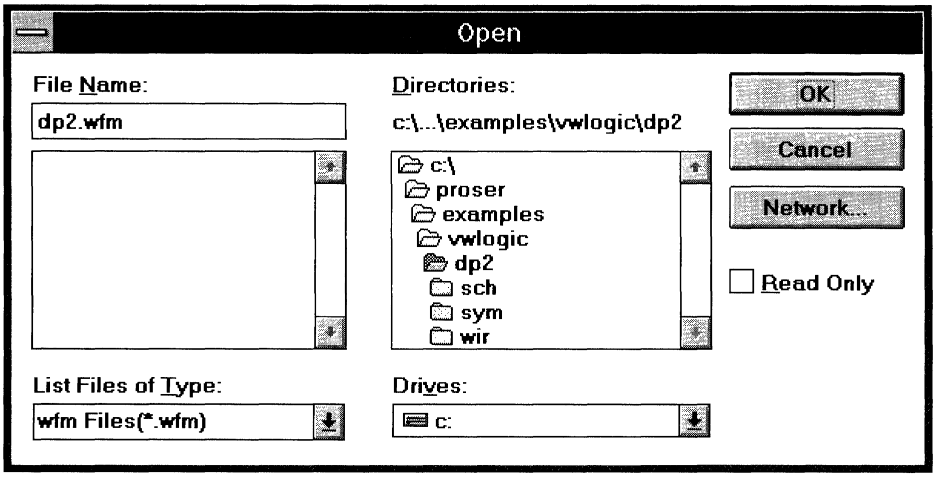


Figure 6-6 Timing Simulation PROwave Icon

The Open dialog box appears, as shown in Figure 6-7.



Open

File Name:

dp2.wfm

Directories:

c:\...\examples\vwlogic\dp2

c:\
proser
examples
vwlogic
dp2
sch
sym
wir

List Files of Type:

wfm Files(*.wfm)

Drives:

c:

Figure 6-7 Open Dialog Box

2. In the File Name list box, select the waveform display file. The name of the waveform file is the name you used when you added the signals to the waveform. The default name is *design*.wfm.

   Be sure to select the file that you specified when you added the signals; otherwise, the link between PROsim and PROwave will not be established.

3. Click on **OK**.

   The Open dialog box closes, and the specified waveform display file opens.

Note: If you do not select a waveform file from the File Name list box but instead enter a new name, a stimulus file is created. A stimulus file creates a stimulus command file and does not annotate values during simulation.

Comparing Functional and Timing Simulation Files

Any easy way to verify that the block and routing delays have not changed the functionality of your design is to compare the functional waveform to the timing waveform. To do so, you must specify different file names when creating the functional and timing waveform files. For example, when simulating functionality, you can call the file *design*f.wfm. When adding signals to the waveform and when simulating timing, specify the file as *design*t.wfm. For more information, see the "Adding Signals, Buses, and Vectors to the Waveform" section in the "Functional Simulation" chapter.

The following assumes that you have created two separate waveform files, the first called *design*f.wfm, created during functional simulation, and the second called *design*t.wfm, created during timing simulation.

1. As shown in the previous section, invoke PROwave and open the timing waveform file, *design*t.wfm.
2. Once the timing waveform file is open, select the **`File`** → **`Open`** command, and select the functional waveform file, *design*f.wfm.

   By default, PROwave cascades the waveforms, as shown in Figure 6-8.

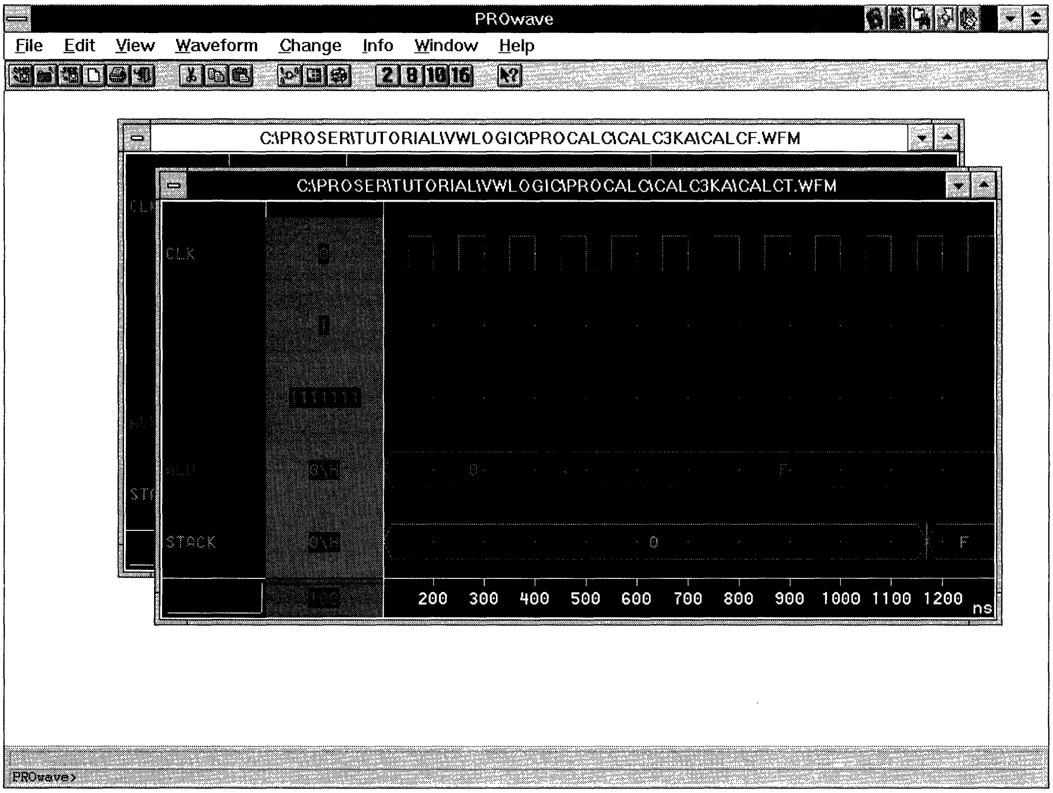

PROwave
File Edit View Waveform Change Info Window Help
C:\PROSER\TUTORIAL\VWLOGIC\PROCALC\CALC3KA\CALCF.WFM
C:\PROSER\TUTORIAL\VWLOGIC\PROCALC\CALC3KA\CALCT.WFM
CLK
ALU
STACK
200 300 400 500 600 700 800 900 1000 1100 1200 ns
PROwave>

Figure 6-8 Cascaded Waveform Windows

To make it easier to compare the two waveforms, you can rearrange the waveform windows.

3. Select the **`Windows`** → **`Tile`** command, which arranges the windows side by side, as Figure 6-9 demonstrates.

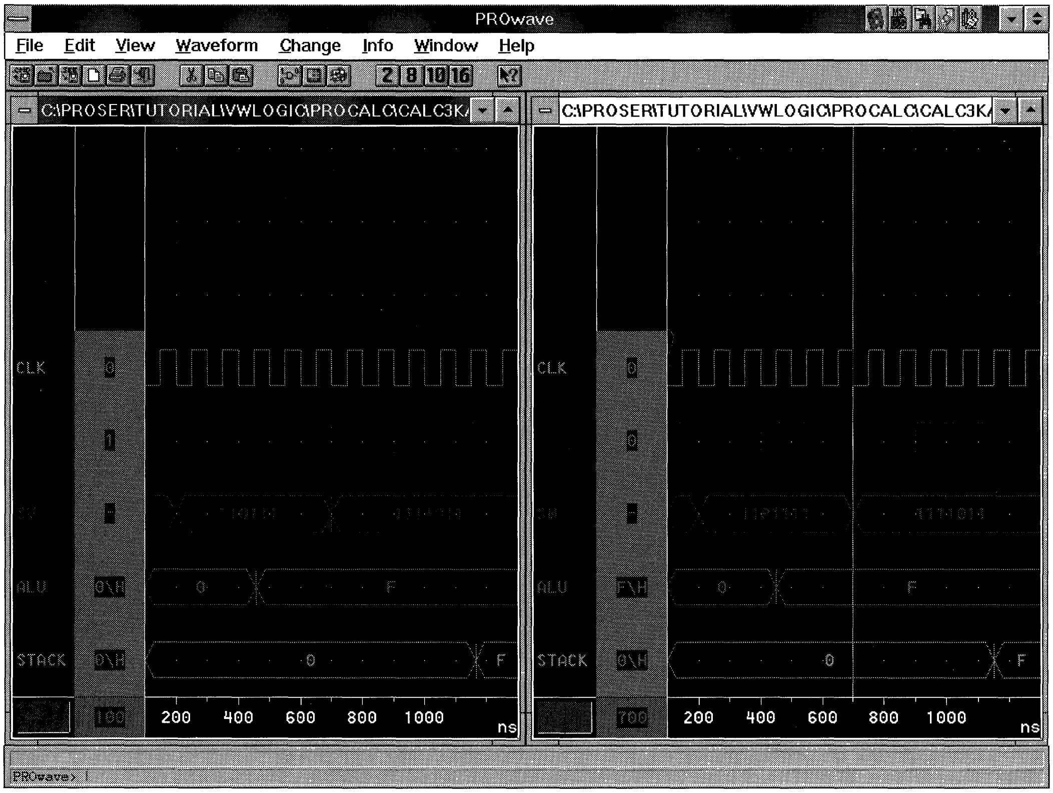

PROwave
File Edit View Waveform Change Info Window Help
C:\PROSER\TUTORIAL\VWLOGIC\PROCALC\CALC3K
CLK
ALU
STACK
200 400 600 800 1000 ns
PROwave>

Figure 6-9 Tiled Waveform Windows

Zooming the Waveform Files

At the initial zoom level, it is sometimes difficult to distinguish differences between waveforms. Zooming in and changing the grid spacing of the values makes inspection of the waveforms much easier.

1. Using the **F9** function key or the **View** → **Region** command, zoom into the hash marks along the bottom of the *design*t.wfm waveform.
2. To change the hash mark spacing, select the **View** → **Grid** → **Space** command.

PROwave prompts for the horizontal step spacing at the PROwave command line.

3. At the `Horizontal step [100]` prompt, type *interval*↵.

   Interval is the tick size. Specifying a number such as 15 for the horizontal step changes the interval of the hash marks to 15 ns.

4. Activate the *design*f.wfm waveform and zoom in to the same region, specifying the same hash mark spacing.

Obtaining a Transition Time

To obtain the exact time of a transition, place the crosshair at the transition in the waveform.

1. Point the mouse at the vertical transition in the desired waveform.
2. Double-click the left mouse button.

   The crosshair is placed at the transition, and the PROwave status bar is updated with the exact transition time.

Obtaining a Delta Time

You can obtain a delta time between two transitions by double-clicking on the first transition, then double-clicking on the second transition. The delta time is reported in the PROwave status bar located above the PROwave command line.

Viewlogic Interface Guide

Design and Simulation Techniques

Chapter 7

Design and Simulation Techniques

This chapter discusses aspects of schematic entry and simulation with PRO Series and Powerview that you should be familiar with to use them effectively.

Xilinx Libraries

The Unified Libraries conform to standards set for the appearance, function, and naming conventions of the library elements. This standardization allows you to convert easily from one Xilinx architecture to another. The primitives and macros in the Unified Libraries should be used to create new designs; creating new designs from previous libraries is not recommended because you cannot mix components from the old libraries and the Unified Libraries in the same design. Refer to the Xilinx *Libraries Guide* for detailed information on the Xilinx libraries.

Primitives and Macros

The Unified Libraries contain three types of components: primitives, soft macros, and relationally placed macros (RPMs). Primitives are those symbols recognized directly by the implementation software such as pads, gates, latches, flip-flops, buffers, and oscillators. Soft macros are schematics that contain primitives and other soft macros. Soft macros have pre-defined functionality but have flexible mapping, placement, and routing. RPMs, available for XC4000 and XC5200 devices, are soft macros that contain both placement information and logic elements.

Note: User-generated hard macros from libraries created before the release of the Unified Libraries must be converted to RPMs with the HM2RPM program. Refer to the "HM2RPM" chapter of the *Development System Reference Guide* for detailed information on hard macro conversion.

X-BLOX

The X-BLOX library contains module generators that describe a system using high-level functions instead of gate primitives. The X-BLOX synthesis tool processes these modules. The X-BLOX library can be used only with XC3000A/L, XC3100A, XC4000, and XC5200 FPGA designs.

The X-BLOX library is supplied in the Viewlogic interface and libraries software, but the X-BLOX software required for processing designs with X-BLOX symbols is not. The X-BLOX software is included in the standard and extended solutions packages and is also sold separately.

User-Created Libraries

You may find it convenient to create your own libraries of frequently used components, especially if you wish to share parts of your designs with other people on your system.

You can make new library components either by editing the soft macros provided by Xilinx and saving them under another name, or simply by saving schematic blocks of your own design.

To create your own library on either PCs or workstations, follow these steps.

1. Enter a schematic for the component.
2. Make a symbol for the component. See the "Design Entry" chapter for instructions on creating symbols.
3. Create a directory for the library. You can place it anywhere on the system. For example, type the following:

   ```
   mkdir \userlib
   ```

   On a workstation, type the following:

   ```
   mkdir /berkeley/userlib
   ```

4. Create three subdirectories: sch, wir, and sym. An example on PCs follows:

   ```
   mkdir \userlib\sch
   mkdir \userlib\wir
   mkdir \userlib\sym
   ```

   Here is an example on workstations:

   ```
   mkdir /berkeley/userlib/sch
   mkdir /berkeley/userlib/wir
   mkdir /berkeley/userlib/sym
   ```

5. Place the SCH, WIR, and SYM files for your component into the library subdirectories, as shown in this example for PCs:

   ```
   copy sch\mycomp.1 \userlib\sch
   copy wir\mycomp.1 \userlib\wir
   copy sym\mycomp.1 \userlib\sym
   ```

   Following is an example for workstations:

   ```
   copy sch/mycomp.1 /berkeley/userlib/sch
   copy wir/mycomp.1 /berkeley/userlib/wir
   copy sym/mycomp.1 /berkeley/userlib/sym
   ```

6. On the PC only, you can optionally use the Setup command in the \proser directory to compress the library into megafile format. However, to make changes to your library, you must skip this step until all changes are complete. The megafile format is read-only.

7. Using the Library List Editor on PCs or a text editor on workstations, edit your viewdraw.ini file to include the new library on your search path. Place your library after the Xilinx libraries *but before the builtin library.* For this example, add the following line:

   ```
   DIR [r] \userlib (userlib)
   ```

   If you choose to compress your library into megafile format, you must specify [m] instead of [r].

You can now specify any component from your user library and place it in your schematic as you would place a Xilinx component.

If you create components for your library that reference other components from your library, make sure that the alias of the first component that you create is correct when you place it. If the alias is not present, the software cannot locate the component at a later time. In other words, you may have to create your library in stages, moving each new component to the library as it is completed.

To verify each new element, remove the SCH, WIR, and SYM files of the component from the subdirectories of your project directory. Place each component in a test schematic and make sure that the symbol appears and the alias on the symbol corresponds to your new library.

Merging Non-Schematic-Based Modules

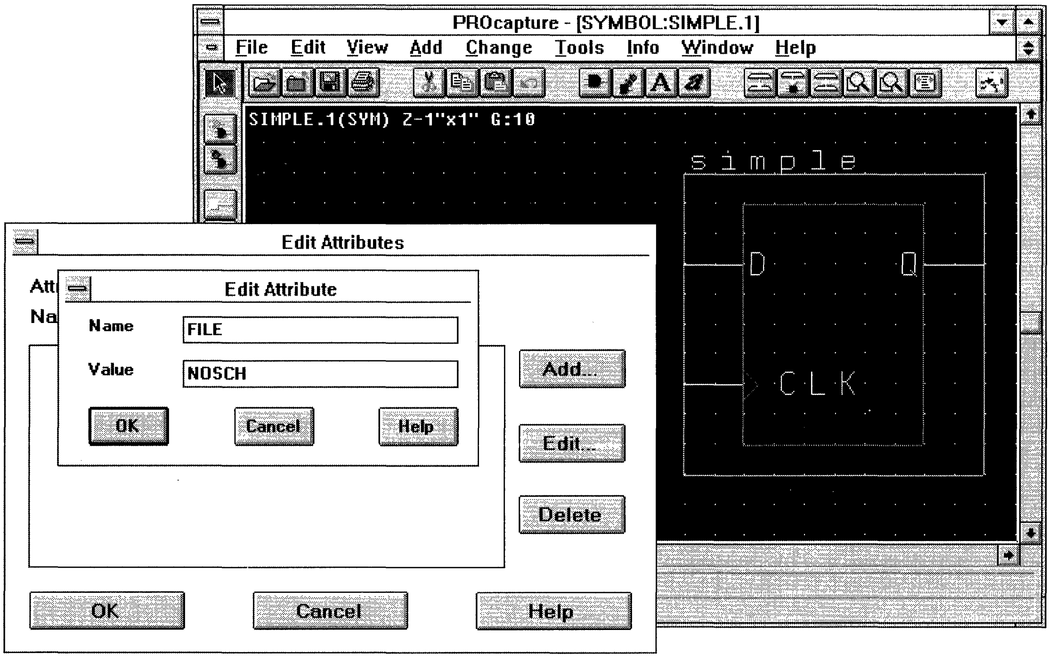
This section describes how to incorporate a symbol representing XNF files for FPGAs and EPLDs or behavior modules for EPLDs into your schematic.

XNF Files

Suppose that you choose to include an XNF file that you have generated using software from one of Xilinx's Alliance partners. You have an XNF file representing a portion of your design. To incorporate this netlist into your schematic, simply create a symbol for the XNF file and place it on your schematic as you would any other component. However, there are two extra requirements that must be observed.

- The symbol must be of type Module. The default block type for a symbol is Composite, which means that there is a schematic for the symbol. You can change the symbol type by selecting Change → Symbol Type → Module. The easiest way to verify the block type is to write the symbol using the File → Save command. The block type is reported on the screen as part of the Tools → Check command output.
- You must specify a FILE=*filename* attribute to define the name of the XNF file referenced by the symbol. For example, if the name of the XNF file referenced by the symbol is myfile.xnf, specify the FILE=myfile attribute. This file must be located in the project directory. You can add the attribute either to the symbol or to each instance of the component after you place it on the schematic. It is recommended that you add the attribute to the symbol itself in the symbol editor, unless you plan to use the symbol to represent several different XNF files with identical pins. In either case, make sure nothing is selected, then select Change → Object Attributes → Dialog. Add the attribute and value to the dialog box. Figure 7-1 shows the addition of the FILE attribute to a symbol.

The "Manual Translation" chapter gives specific instructions for creating certain types of symbols, such as Xilinx ABEL and X-BLOX symbols. Although given in the context of simulation, these instructions are valid for other applications as well.


PROcapture - [SYMBOL:SIMPLE.1]
File Edit View Add Change Tools Info Window Help
SIMPLE.1(SYM) Z-1"x1" G:10
simple
D
Q
CLK
Edit Attributes
Edit Attribute
Add...
Edit...
Delete
OK
Cancel
Help

Name: FILE
Value: NOSCH

Figure 7-1 Adding a FILE Attribute to a Symbol

XEPLD Behavioral Modules

Behavioral modules are functionally defined using a text-based behavioral description file. Behavioral descriptions may be in the form of Boolean equations, such as ABEL, or in the form of a hardware description language (HDL). There are several PLD compilers and HDL synthesizers that support Xilinx EPLDs.

You may want to use a behavioral module in your design for one of the following reasons.

- If a portion of your design is already implemented using a lower-density programmable logic device (PLD), you can re-use your existing PLD equations without having to redraw the same logic schematically.
- Some complex combinational logic functions, like state machines, are more conveniently expressed using a behavioral language than schematically.

Each behavioral description file must first be compiled before you can implement your schematic design. Run the appropriate PLD compiler or synthesizer, for example, XABEL, to read your behavioral description file. Be sure to specify Xilinx EPLDs as the target technology. The compiler produces either a PLUSASM-language PLD equation file or an XNF-formatted netlist. If you expressed your behavioral description directly in the PLUSASM equation language, you need not perform any compilation before implementing your schematic.

If your compiler produces an XNF-formatted netlist, refer to the "XNF Files" section for custom symbol special requirements.

If your compiler produces a PLD equation file (or you created your own PLUSASM PLD file), you can use the Symbol Generation utility in the Windows program group to automatically generate a Viewlogic symbol based on the input/output signals defined in your behavioral description file.

1. After compiling your PLD file, invoke the Symbol Generation Utility icon.
2. In the dialog box that appears, enter the name of the PLD file, or go to the Browse menu and select **`PAL Equation Files`** as the file type to get a listing.
3. Then select **`Viewlogic`** as the symbol type.

The utility software reads your PLD file, checking for errors, then generates a Viewlogic symbol file, SYM\*filename*.1, where *filename* is the name of your PLD file.

You can create and store your custom symbols and their behavioral description files in your local project directory, or you can create a custom library directory to store your custom symbols so they can be used in multiple projects. You should never add custom symbols to the EPLD library directory or modify any of the library symbols. You can, however, copy any of the Xilinx-supplied symbols from the library into your project or custom library directory and modify them to suit your design needs.

When using behavioral modules in your design, you must link each instance of your custom symbol in your schematic to the associated behavioral description file.

If you are using XACT*step* Version 6.0, attach a FILE=*filename* attribute to each behavioral module instance in your schematic, where *filename* is the name of the PLD equation file produced by the PLD compiler or synthesizer. The file must be located in the project directory.

You must also attach a DEF=PLD attribute to each behavioral description instance in your schematic. DEF=PLD tells the implementation software to read a PLUSASM equation file and automatically translate it into the required XNF netlist before including the logic into your design.

If you are using XACT*step* Version 5.2, attach a PLD=*filename* attribute to each behavioral module instance in your schematic, where *filename* is the name of the PLD equation file produced by the PLD compiler or synthesizer. This file must be located in the project directory.

Attributes

Attributes are instructions placed on symbols or nets in an FPGA or EPLD schematic that allow you to control aspects of software processing. Attributes express information specific to each design, as opposed to run-time options entered in the Design Manager.

Most attributes are described in detail in the Xilinx *Libraries Guide*. EPLD attributes are described in detail in the *XEPLD Schematic Design Guide*.

PINTYPE

When creating a symbol, you may want to specify the directionality of the pins by using the PINTYPE attributes listed in the following tables. You may also want to add PINTYPE attributes if you would like more extensive connectivity checking performed on your schematic.

Note: It is not required that you add PINTYPE attributes to a user-created symbol unless the symbol represents an XNF file to be included in the design. The Xilinx primitives and macros already have PINTYPE attributes on them and should be sufficient for error checking.

The syntax of the PINTYPE attribute is the following:

PINTYPE=*attribute*

where *attribute* can be one of the following in the left-hand column:

IN	Input
OUT	Output
BI	Bidirectional
TRI	3-state
CHIPIN	Xilinx external input signal
CHIPOUT	Xilinx external output signal
CHIPBI	Xilinx external bidirectional signal
CHIPTRI	Xilinx external 3-state signal
OCL	Open collector
OEM	Open emitter

If the PINTYPEs are specified, the matrices shown in Table 7-1 and Table 7-2 are used to determine valid PINTYPE connections. Pin-to-pin connections are connections between pins on the same level of hierarchy. Pin-to-block connections are connections between external signals on a lower level of hierarchy and the corresponding symbol pins above.

Table 7-1 Pin-to-Pin Connection Matrix

		PIN TYPES ON SYMBOL PINS									
		IN	**OUT**	**BI**	**TRI**	**OEM**	**OCL**	**CHIP IN**	**CHIP OUT**	**CHIP TRI**	**CHIP BI**
PINS IN BLOCKS	**IN**	OK	OK	W2	OK	OK	W1	ERR	ERR	ERR	ERR
	OUT	OK	ERR	ERR	ERR	W4	ERR	ERR	ERR	ERR	ERR
	BI	W2	ERR	W2	OK	W3	W1	ERR	ERR	ERR	ERR
	TRI	OK	ERR	OK	OK	OK	W1	ERR	ERR	ERR	ERR
	OEM	OK	W4	W3	OK	OK	OK	OK	OK	OK	OK
	OCL	W1	ERR	W1	W1	OK	W1	ERR	ERR	ERR	ERR
	CHIPIN	ERR	ERR	ERR	ERR	OK	ERR	ERR	OK	OK	ERR
	CHIPOUT	ERR	ERR	ERR	ERR	ERR	ERR	OK	ERR	ERR	ERR
	CHIPTRI	ERR	ERR	ERR	ERR	OK	ERR	OK	ERR	ERR	ERR
	CHIPBI	ERR	ERR	ERR	ERR	OK	ERR	ERR	ERR	ERR	OK

In this table, the abbreviations have the following meanings.

W1 = Warning 1 — The pin attribute must have an OEM on the same net somewhere in the hierarchy; if it does not, the software issues a warning.

W2 = Warning 2 — The pin attribute must have either TRI, or OCL and OEM on the same net somewhere in the hierarchy; if it does not, the software issues a warning.

W3 = Warning 3 — The pin attribute must have either TRI or OCL on the same net somewhere in the hierarchy; if it does not, the software issues a warning.

W4 = Warning 4 — OEM and OUT are on the same net; OEM has no effect.

Table 7-2 Pin-to-Block Connection Matrix

PIN TYPES ON NETS WITHIN BLOCKS	PIN TYPES ON BLOCKS									
	IN	OUT	BI	TRI	OEM	OCL	CHIP IN	CHIP OUT	CHIP TRI	CHIP BI
IN	OK	W1	W2	W4	ERR	W5	ERR	ERR	ERR	ERR
OUT	ERR	OK	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
BI	ERR	ERR	OK	OK	ERR	W5	ERR	ERR	ERR	ERR
TRI	ERR	OK	OK	OK	ERR	ERR	ERR	ERR	ERR	ERR
OEM	ERR	ERR	W2	W4	ERR	OK	OK	OK	OK	OK
OCL	ERR	ERR	W3	W3	ERR	OK	ERR	ERR	ERR	ERR
CHIPIN	ERR	ERR	ERR	ERR	ERR	ERR	OK	ERR	ERR	OK
CHIPOUT	ERR	ERR	ERR	ERR	ERR	ERR	ERR	OK	ERR	OK
CHIPTRI	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	OK	OK
CHIPBI	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	OK

This table assumes that OEM is only used for pull-up and pull-down resistors. It also assumes that OCL is both open collector and open drain. In this table, the abbreviations have the following meanings.

W1 = Warning 1 — The pin attribute must have an OUT on the same net somewhere in the hierarchy, within this block or below; if it does not, the software issues a warning.

W2 = Warning 2 — The pin attribute must have one or more of BI or TRI, or OCL and OEM on the same net anywhere in the hierarchy; if it does not, the software issues a warning.

W3 = Warning 3 — The pin attribute must have an OEM on the same net somewhere in the hierarchy; if it does not, the software issues a warning.

W4 = Warning 4 — The pin attribute must have one or more of BI or TRI, or OCL and OEM somewhere in the hierarchy, within this block or below; if it does not, the software issues a warning.

W5 = Warning 5 — The pin attribute must have an OCL and can have an OEM somewhere in the hierarchy, within this block or below; if it does not, the software issues a warning.

The CHIP*xxx* attributes should only be used if an I/O symbol primitive is placed on a lower level of hierarchy.

Power and Ground Signals

Unused inputs on symbols should not be left unconnected. You should never assume a default value for any unconnected symbol input except basic logic gates such as AND or OR. In some cases, if a control input to a library symbol is left unconnected, the resulting behavior may be different than if the input were tied either High or Low. For example, if the CE input of an FDRE component is left unconnected, the CE logic that selects between the D-input and Q-feedback would be removed and the flip-flop would load the value of the D-input ORed with its Q-feedback, clearly not the intended functionality. Both functional and timing simulation will exhibit this resulting incorrect behavior.

Unused inputs should be tied to a constant High or Low logic level in the schematic. Use the VCC or GND symbol from the library to tie a net to a constant logic High or Low. As an alternative, you can specify a constant High or Low value by connecting a net on the component input pin and then labeling the net as "VCC" or "GND," which is recognized by the Xilinx software.

EPLD Design Issues

This section describes aspects of design entry that are unique to EPLDs.

Inputs, Outputs, and Bidirectionals

Many of the components in the EPLD library have special features that take advantage of EPLD architecture. The following sections describe some of these features.

Clocks and Global Control Nets

The EPLD fitter automatically uses the global clock and global output enable nets of the device whenever possible. If you need to control the use of global resources explicitly, there are special-purpose buffer symbols, BUFG and BUFFOE, in the library that you can use instead of an IBUF.

The fitter automatically uses input-pad registers in the device to implement simple flip-flops that you connect directly to input buffers (IBUF) in your design, if possible. If you need to control the use of input-pad registers and latches explicitly, use one of the IFD or ILD symbols in place of the IBUF and FD-type flip-flops, as shown in Figure 7-2.

Note: Do not connect an IBUF to the D input of an IFD/ILD symbol.

Refer to the Xilinx *Libraries Guide* for specific application rules for each symbol.

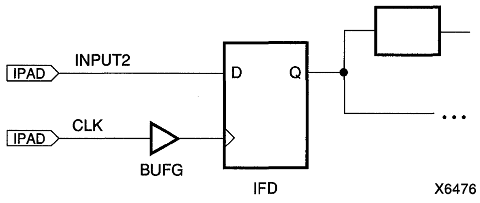


Figure 7-2 Input Buffers and Input-Pad Flip-flops

The fitter looks for opportunities to automatically assign tristate enable signals to the EPLD's global fast output enable (FOE) lines. If you want to take advantage of an FOE global line explicitly, use a BUFFOE input buffer instead of an IBUF, and connect it to an OBUFEX1 output buffer instead of OBUFE, as shown in Figure 7-3.

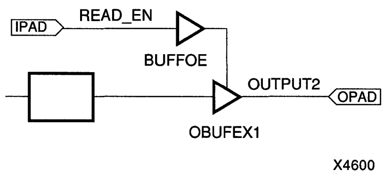


Figure 7-3 Using a Global FOE Explicitly

Note: You should always label all the nets connecting PAD symbols and input/output buffer symbols. These names are used by the software to refer to your device pins in the reports and during simulation.

To represent a bidirectional I/O pin, use an IOPAD symbol connected to both the input of an IBUF or IFD/ILD and the output of an OBUFE or OBUFT, as shown in Figure 7-4.

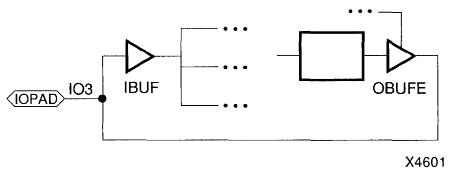


Figure 7-4 Bidirectional I/O Pin

Output and Tristate Buffers

If a signal going into a common output buffer (OBUF) is generated by any component containing a tristate buffer (like BUFE or a behavioral module), the tristate control signal is used to enable and disable the device output pin driver. This behavior is unique to EPLDs and is not reproduced in FPGAs. It is shown in Figure 7-5.

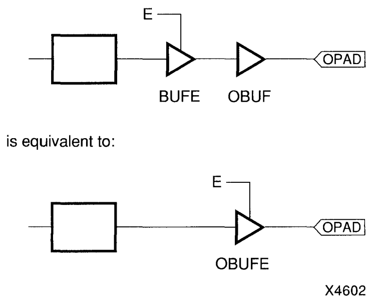


Figure 7-5 Output Enable Behavior in EPLDs

If you use a behavioral module in your schematic and connect one of its outputs to an output buffer like OBUF, you can control the EPLD device output pin using a tristate control equation in the behavioral module, as shown in Figure 7-6.

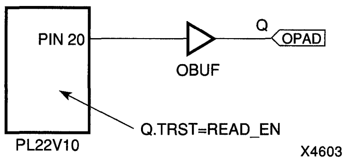


Figure 7-6 Controlling Tristate Output Using a Behavioral Module

If you want to use a tristate behavioral module output to control a bidirectional I/O pin of the EPLD, connect the OBUF output to an IOPAD and IBUF or IFD/ILD. If the same behavioral module symbol that generates the output is also to receive the I/O pin input, you must use a separate pin of the behavioral module's symbol to receive the signal from the IBUF. Do not tie the signal received from an IBUF to the net driving the OBUF of the same IOPAD, as shown in Figure 7-7; these input and output nets must remain separate, as shown in Figure 7-8.

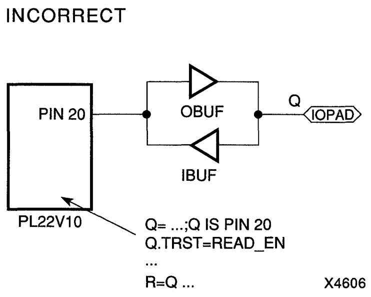


Figure 7-7 Incorrect Way to Connect a Bidirectional Pin

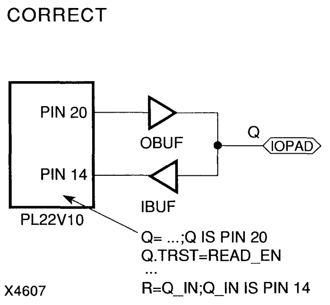


Figure 7-8 Correct Way to Connect a Bidirectional Pin

If your design calls for tristate multiplexing of multiple output sources, it is best to output each signal source on its own set of tristate output pins and tie the signals together off-chip. You cannot connect more than one signal source to the same OBUF or OPAD, as shown in Figure 7-9 and Figure 7-10.

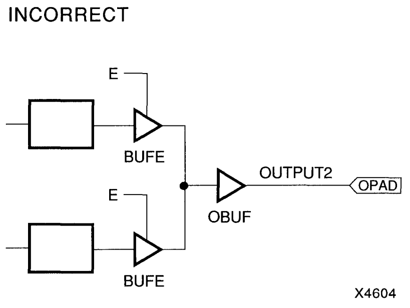


Figure 7-9 Incorrect Tristate Multiplexing of EPLD Outputs

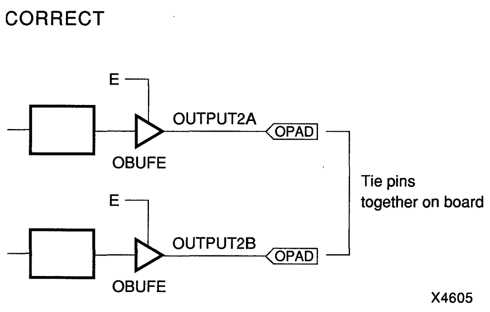


Figure 7-10 Correct Off-Chip Tristate Multiplexing of EPLD Outputs

On-Chip Tristate Multiplexing

EPLD components emulate tristate signals internally by gating the macrocell feedback to the interconnect matrix. (Macrocell feedback signals are never physically in a high-impedance state.) You can tie together the outputs of multiple tristate buffer symbols, like BUFE, BUFT, or tristate behavioral module outputs, to multiplex these signals on-chip, as in Figure 7-11. You cannot connect such tied signals to an output buffer; instead you must pass a tied signal through a logic symbol, like BUF, before driving an output port.

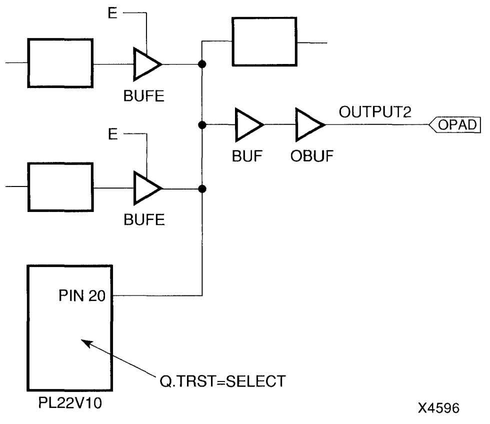


Figure 7-11 Correct On-Chip 3-State Multiplexing

EPLD-Specific Components

EPLD-specific components are symbols that take advantage of unique EPLD architecture features and that are not found in any of the other Xilinx family libraries. Some EPLD-specific components perform functions similar to common library symbols but require a different set of symbol pins to access the special EPLD features. In general, you should use EPLD-specific components only when a particular logic or I/O function in your design would benefit from the special advantage offered by the component. The EPLD-specific library symbols are listed in Table 7-3.

Table 7-3 Common and EPLD-Specific Symbols

Common Symbols	EPLD-Specific Symbols
Accumulators	
ACC1	ACC1X1 or ACC1X2
ACC4	ACC4X1 or ACC4X2
ACC8	ACC8X1 or ACC8X2
ACC16	ACC16X1 or ACC16X2
Adders	

Common Symbols	EPLD-Specific Symbols
ADD1	ADD1X1 or ADD1X2
ADD4	ADD4X1 or ADD4X2
ADD8	ADD8X1 or ADD8X2
ADD16	ADD16X1 or ADD16X2
ADSU1	ADSU1X1 or ADSU1X2
ADSU4	ADSU4X1 or ADSU4X2
ADSU8	ADSU8X1 or ADSU8X2
ADSU16	ADSU16X1 or ADSU16X2
Counters	
CB2CLED	CB2X1
CB4CLED	CB4X1
CB8CLED	CB8X1
CB16CLED	CB16X1
Input Buffers	
IBUF	BUFFOE or BUFCE
Input Registers	
IFD	IFDX1
IFD4	IFD4X1
IFD8	IFD8X1
IFD16	IFD16X1
Output Buffers	
OBUFE	OBUFEX1

If, however, you want your design to remain re-targetable to either an EPLD or FPGA device, use only the common components. Most EPLD-specific components have at least one common counterpart.

Note: Each common component functions identically in EPLD and FPGA devices. However, there may be a significant difference in efficiency or performance between the families. Whenever you map a design to a new device, you should do the following.

- Check the reports created during integration carefully to make sure that the way you expressed your design does not consume excessive logic resources of the target device.
- Perform timing analysis or simulation to catch any inefficient parts of the design.

If you need further information on XEPLD software and design techniques, see the *XEPLD Schematic Design Guide.*

Counters

Up/down counters in the common library, such as CB8CLED, have a single CEO output, which changes in response to the up/down direction input. Gating of the CEO function like this does not allow it to be optimized for zero delay, so if common up/down counters are cascaded, they cannot operate at their maximum original frequency.

The EPLD-specific up/down counters such as CB8X2 have separate outputs for the up and down directions, CEOU and CEOD, that can be optimized. You can cascade these components without impacting their maximum frequency.

Arithmetic Components

The ADD-, ADSU-, and ACC-type common library components use the EPLD macrocell carry chain between output bits within the same component, but not for cascading. If you cascade these components, the carry signals (CI and CO) go through the UIM and incur a delay. The CI and CO pins can connect to any ordinary logic components or I/O ports, but not to the CI and CO pins of EPLD-specific arithmetic components.

The EPLD-specific adders, whose names end in X1 or X2, use the EPLD macrocell carry chain for cascading without incurring a UIM delay. Their CI and CO pins can only be connected to the CI and CO pins of another EPLD-specific arithmetic component.

Device-Specific Components

Some of the components in the EPLD library are not supported by all EPLD devices. For example, the following components require features only present in devices containing high-density function blocks and are therefore not supported by the XC7318 or XC7336 devices:

- ADD, ADSU, and ACC symbols
- BUFCE
- IFD, IFDX1, and ILD symbols
- COMPM symbols
- LD symbols
- XOR7, XOR8, and XOR9

Please refer to the Xilinx *Libraries Guide* for details.

FPGA Design Issues

Although there are a few differences between FPGA designs and other ASIC or board-level designs, FPGA schematic design generally involves the same techniques as designs for other technologies. Most of these differences involve adding FPGA-specific information to the schematic. The information described in this section on attributes, library primitives, X-BLOX symbols, and MemGen symbols is used by the FPGA design implementation software during placement and routing of your design.

Using MemGen

The Xilinx memory compiler, MemGen, provides an easy way to create RAM and ROM memories within Xilinx XC4000 FPGAs. With MemGen, you can automatically create memories ranging from one to 32 bits wide and up to 256 words deep. The MemGen program is described in the *Xilinx Reference Guide.*

Invoke MemGen with the file name to be used for the memory, as in the following example:

```
memgen shifter -v
```

MemGen creates a memory definition file called shifter.mem, an XNF file to represent it, and a Viewlogic symbol. Use the Add → Component command to place the memory symbol on a schematic. See the "Merging Non-Schematic-Based Modules" section for a detailed description of this process.

Figure 7-12 is an example of a shifter.mem file.

```
; ============================================================
; example.mem:  A 32-word deep by 16-bit wide ROM memory.
; ============================================================
;
TYPE  ROM     ; The memory is a ROM
DEPTH  32     ; The memory is 32 words deep
WIDTH  16     ; Each memory word is 16 bits wide
;
SYMBOL VIEWLOGIC BUS ; Build a VIEWLOGIC symbol with bus
inputs
;
DEFAULT 0     ; <-- Default value for unspecified locations

DATA    2#1111_0000_1111_0000#,    ; Binary data
        8#17777#,                  ; Octal data
        10#23#,                    ; Decimal data
        16#4A#,                    ; Hex data
        4F                         ; Unspecified base assumed
                                   ; to be Hex

; ============================================================
;  Your ROM memory uses approximately 128 two-input NAND
;  gates as measured by:
;
;  GATES = (WIDTH * DEPTH)  <-- The average cost for the ROM
;  function.
```

Figure 7-12 Sample Memory Definition File

Using X-BLOX

You can use X-BLOX modules in your Viewlogic design. The installation program places the X-BLOX library with the other Xilinx libraries. You can use the other libraries available in these directories, either the XC3000, XC4000, or the XC5200 library with the X-BLOX library.

Add the following line to your viewdraw.ini file on a PC:

```
DIR [m] \proser\unified\xblox (xblox)
```

Add the following line to your viewdraw.ini file on a workstation:

```
DIR [m] /powerview_path/unified/xblox (xblox)
```

To retrieve a library component, use the Add → Component command and enter the name of the X-BLOX component that you need.

Adding Attributes

To add attributes to an X-BLOX component, find the names and the proper values of the attributes in the *X-BLOX User Guide*. In ViewDraw, use either the Add → Attribute or the Change → Attribute → Dialog → All command to specify attributes; in PRO Series, use the Add → Object Attribute or the Change → Object Attributes → Dialog command to specify them.

Interfacing X-BLOX and PROcapture or ViewDraw Buses

Special components are required to interface X-BLOX buses with standard PROcapture or ViewDraw buses. An X-BLOX bus is not the same as a bus normally used in PROcapture or ViewDraw. The width of an X-BLOX bus is not defined by the name attached to the bus. In fact, X-BLOX buses must never be given indexed names such as OUT[7:0], because the bus pins on X-BLOX symbols do not have indexed names. All X-BLOX symbols have unindexed bus pins so that the same symbol can be used in any design, regardless of the width of the buses in the design. If an indexed bus is attached to one of these unindexed bus pins, PROcapture or ViewDraw flags an error on the bus. BUS_IF symbols are needed as interfaces between X-BLOX buses and normal Viewlogic buses. The specific BUS_IF symbol required depends upon the width of the bus being interfaced. The BUS_IF symbols are called BUS_IF*nn*, where *nn* is the number of bits in the bus.

Note: X-BLOX bus labels must end with a letter, not a number.

Processing the Design

Once an XNF file is generated, follow the procedure outlined in the "X-BLOX Tutorial" chapter in the *Viewlogic Tutorials* manual to process X-BLOX modules for PRO Series. XMake recognizes X-BLOX designs and automatically processes them using the X-BLOX flow.

Simulating Oscillators

The OSC symbol corresponds to the oscillator driver on the XC2000 and XC3000 FPGA devices. The oscillator driver is a special dedicated circuit on the FPGA that can drive an off-chip crystal oscillator to generate a high-speed clock signal in the FPGA. When the oscillator is used, the two dedicated IOBs, XTAL1 and XTAL2, are reserved and cannot be used for any other purpose. The XTAL1 and XTAL2 pin numbers depend on the part and package type and are listed in *The Programmable Logic Data Book.*

The OSC oscillator for the XC2000 and XC3000 families can only be used to drive the input of the ACLK. No other connection is allowed. The OSC and ACLK symbols are combined into a single symbol, GXTL, which you can place on your schematic instead of OSC and ACLK.

OSC4 is an internal five-frequency clock signal generator for XC4000 parts. It provides a clock of approximately 8 MHz and any two of 500 KHz, 16 KHz, 490 Hz, and 15 Hz. Frequencies vary due to process variations.

Simulating OSC and GXTL (XC2000 and XC3000 Only)

You can simulate the OSC oscillator in PROsim by applying the clock signal to the ACLK connected to the OSC symbol. To simulate the GXTL oscillator, apply the clock signal to the ACLK within the GXTL symbol. In either case, this step forces the clock waveforms onto the net output from the OSC or GXTL symbols. For information on how to create a clock source during simulation, see the "Functional Simulation" chapter.

Simulating OSC4 (XC4000)

To simulate the outputs of the OSC4 component, apply clock sources to the nets connected to the outputs being used.

Hold Violations

During simulation, you may encounter hold violations on XC3000 flip-flops used in your design. According to *The Programmable Logic Data Book,* there is no hold time for the flip-flops in a CLB unless the DIN pin is used. The absence of hold-time requirements is reflected in the modeling in the routed XNF file similar to the following.

```
SYM, Q1.QX, DFF, INIT=R
  PIN, Q, 0, Q1.QX, 5.0
  PIN, D, I, Q1.F
  PIN, C, I, CAB, 6.1
  PIN, CE, I, BCE, 0
  PULSE, C, +, 4.0
  SETUP, D, C, +, 0.0, 4.5
END
```

The Setup line indicates that the D input pin is clocked by C, which is positive-edge-triggered, and has a setup time of 0 ns and a hold time of 4.5 ns. This XNF file is for an XC4000 device with a speed grade of -5. If you look up the setup time for a CLB flip-flop in *The Programmable Logic Data Book,* the setup time is 4.5 ns, and the hold time is 0 ns.

The setup and hold-time specifications in the Xilinx data sheets are based on a comparison of the CLB input signal and the CLB clock input, which is also the clock input of the flip-flop. This comparison is illustrated as (a) in Figure 7-4. Before PROsim reads your netlist, the CLB is broken down into gates and flip-flops. PROsim checks the setup and hold time by comparing the D input of the flip-flop and the clock input, shown as comparison (b) in Figure 7-4.

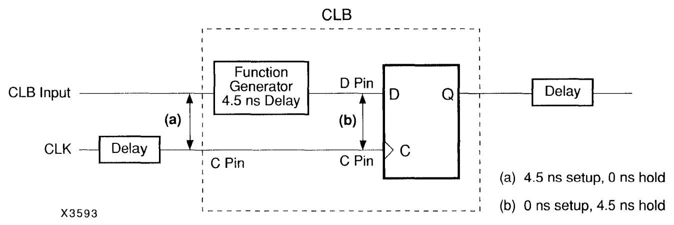

CLB
CLB Input
Function Generator 4.5 ns Delay
D Pin
D
Q
Delay
(a)
(b)
CLK
Delay
C Pin
C Pin
C
(a) 4.5 ns setup, 0 ns hold
(b) 0 ns setup, 4.5 ns hold
X3593

Figure 7-4 Hold Violation Example Circuit

Consequently, PROsim may report what appears to be a setup violation from the viewpoint of the CLB specification as a hold violation. A hold violation waveform is shown in Figure 7-5. Since the sum of the setup and hold-time requirements is the same in either case, it does not matter whether the violation is reported as a setup or hold-time violation.

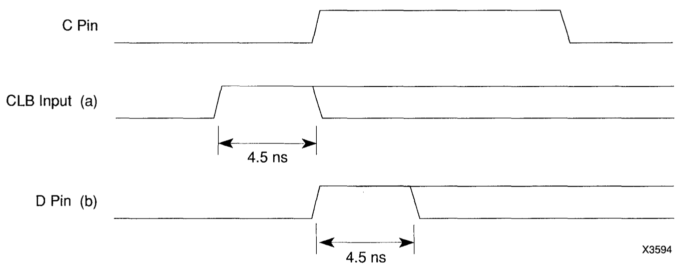


Figure 7-5 Hold Violation Waveform Output

Simulating EPLD Designs

To use PROsim effectively to simulate EPLD designs, you must understand how to initialize the device, manipulate nodes during simulation, and interpret back-annotated simulation results in the schematic. These topics are discussed in the following sections.

Initialization Using PRLD and MRESET

During functional and timing simulations, the PRLD signal initializes the device flip-flops. To better model the device, the tools also insert the MRESET pin, an active-Low reset, into the simulation netlist. You can use this signal for device simulation by placing a pin named MRESET on the top-level symbol. The MRESET pin is connected through an inverter to the global PRLD signal; either can be used to initialize the circuit.

Effects of EPLD Optimization on Timing Simulation

The PROsim simulator allows you to view and force logic values on most nodes (nets) in your design. If you are simulating functionally, all nets in your original design are visible.

However, if you are performing a timing simulation, internal nodes connected to components that are optimized are not observable if the component to which the node is connected was removed during optimization. Components "disappear" when optimized into other components or the universal interconnect matrix (UIM). Input registers may also be optimized into input pads. Generally, nets that do not originate from macrocell outputs or I/O buffers are not visible during timing simulation.

When optimization removes a component, the nets associated with it are removed also and therefore cannot be observed or manipulated during timing simulation.

When you view your design schematic in the PROcapture window, these nets are back-annotated with a "?" during simulation, indicating that the associated net name is undefined in the simulation model. If you must restore a node's visibility to debug your design, you can temporarily connect the node to an output buffer (OBUF).

Node Visibility During EPLD Timing Simulation

Nets between IPADs and IBUFs are observable, as are nets between OBUFs and OPADs. However, the outputs of IBUFs and the inputs of OBUFs are not observable. The outputs of IFDs or registers optimized into input pad registers are also not observable. Other nets may or may not be observable, depending on the results of optimization. To see which of these other nets are observable, see the *design*.lgx file.

If you want to observe the outputs of registers mapped into macrocells, and these nets are not listed in the LGX file, you can view these nets during simulation using the following method. (You can also observe the immediate inputs of these registers, although these are unlikely to be the component inputs.)

1. Label the net connected to the register input or output. (If the node is not optimized, it will be visible by its original name during timing simulation.)
2. Label the register component having the output or input that you want to view.
3. Implement the design.
4. Using a text editor such as Write, view the mapping report (*design*.map). Look for the component label in the Function Name column.
5. When you find the component label, look for the specific output that you want if the component has more than one output; even if you want the input, look for the output. The output name is separated from the component name by a colon (:).

   Inversions are performed before the register in XC7000 devices. If the output name of the register has an _INV suffix, the signal you observe will be inverted from the node in your original design.
6. Look in the Macrocell Location column to find the macrocell number to which the output has been assigned. The Macrocell Location name has the following format.

   FB*fb#***-***mc#*

   Here is an example:

   ```
   FB8-6
   ```
7. For the register output, use the following name as the signal name for timing simulation:

   MCQ_*fb#_mc#*

   An example follows:

   ```
   MCQ_8_6
   ```
8. You can also observe the signal on the D-input to the same register using the name.

MCD_*fb#_mc#*

Following is an example:

```
MCD_8_6
```

Keep in mind that timing simulation simulates your design as it is actually implemented in the chip.

Creating Stimulus Files in PROwave

Editing the waveforms automatically creates a command file of the PROwave waveforms. This command file can be executed during simulation to define input waveforms.

1. Delete all output signals and internal nodes from the waveform view, leaving only the device input signals.

Warning: Any output waveforms that remain in your new command file are forced onto your simulation outputs and may cause false results when you run the file.

2. Use the **`Edit`** → **`Insert`** command in PROwave to edit the input waveforms to produce the desired input stimuli. For more information, consult the Viewlogic documentation.
3. Save your input waveforms in a command file, *stimulus*.cmd, by using the **`File`** → **`Save As`** → **`Cmdfile`** command.

Note: If you want to be able to reload the waveforms and edit them at a later time, use the **`File`** → **`Save As`** → **`Savefile`** command to save the waveforms as a *stimulus*.see file.

4. Edit the *stimulus*.cmd command file and add any desired vectors or other input stimulus.
5. Return to the PROsim window and enter **`restart`**.
6. Run the stimulus file by typing *stimulus*.cmd at the PROsim prompt.
7. If you need to edit your waveforms again, open the *stimulus*.see file in a PROwave window.

Viewlogic Interface Guide

Manual Translation

Manual Translation

The previous chapters focused on using PROflow and the Xilinx Design Manager to process your design. This chapter describes how to manually execute the XSimMake, XMake, and XEMake programs used by PROflow and the Xilinx Design Manager. It also explains each individual program in the entire translation process, so you can apply the appropriate ones if your design has special needs that XSimMake, XMake, or XEMake do not address. This chapter provides the syntax required to execute each translation program from the operating system.

Design Flows

The following figure shows the various design flows involved in manual translation and the different types of files created during each. Flow 1 shows the creation of functional simulation data, flow 2 shows design implementation, and flow 3 shows the creation of timing simulation data.

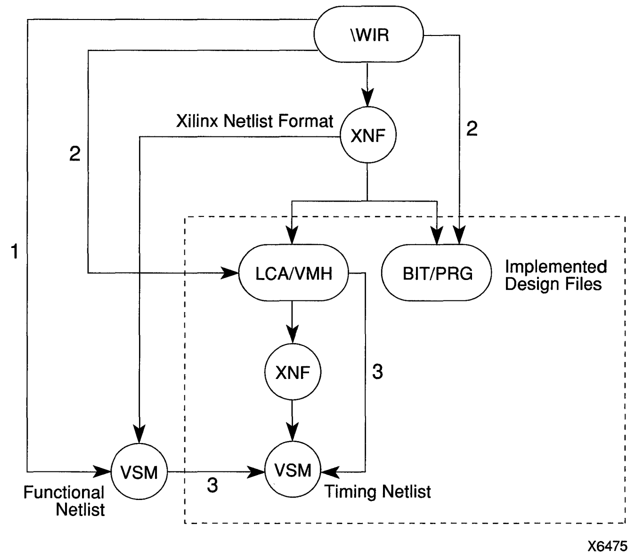


Figure 8-1 Four Types of Manual Translation

Design Procedure

Use the following general steps when creating a design.

1. Enter the design.
2. Verify the design by performing a functional simulation.
3. Modify the design, if necessary, and repeat step 2.
4. Implement the design.
5. Verify the design by performing a timing simulation.
6. Re-implement the design, if necessary, and repeat step 5.
7. Download the final design and verify the hardware.

The processing required for steps 2, 4, and 5 is automatically performed by the XSimMake, XMake, and XEMake tools. XSimMake creates both functional and timing simulation netlists. XMake and XEMake translate design files, creating placed and routed FPGAs and fitted EPLDs, respectively.

Automatic Programs

The first half of this chapter describes the XSimMake, XMake, and XEMake programs, which automatically create simulation data and implement your design.

Using XSimMake to Create Functional Simulation Data

XSimMake automatically runs all the programs and translators required for functional and timing simulation. Not only does it support designs containing only non-X-BLOX schematics, it also allows you to perform functional and timing simulation on designs containing special symbols such as X-BLOX, Xilinx ABEL, and MemGen symbols. It also accepts symbols whose logic is represented with a FILE=*filename* attribute, where *filename* is the name of an XNF file. XSimMake can also initialize MemGen-created ROMs for simulation.

You can run XSimMake from the command line or from Windows.

FPGA Designs

This section describes how to run XSimMake on the command line to perform a functional simulation on FPGA designs.

For FPGAs, XSimMake automatically runs the translation programs required to set up your design for functional simulation. It creates a simulation directory called s*design* ("s" concatenated with your design name) under the design project directory. It creates subdirectories in the simulation directory called sch, wir, and sym; in essence, it makes a user-created library. XSimMake modifies the viewdraw.ini file in the design project directory, adding the new simulation directory to the library list. The alias for the new library is s*design*. This addition to the viewdraw.ini file allows you to access the simulation directory without leaving the project directory and without changing projects. XSimMake writes the simulation network (VSM file) into the project directory, naming it s*design*.vsm.

The directory structure before you run XSimMake resembles the following.

```
design
   /sym
   /sch
   /wir
```

After you run XSimMake, the directory appears as follows:

```
design
   /sym
   /sch
   /wir
/sdesign
   /sym
   /sch
   /wir
```

For portions of the design that do not contain any special components, such as X-BLOX, Xilinx ABEL, or MemGen components, only the top-level symbol is re-created. You do not have to use XSimMake if your design does not contain special components. The "Individual Programs" section of this chapter describes the steps involved in creating VSM files manually. However, Xilinx strongly recommends that you run XSimMake instead of translating manually.

The new simulation schematic generated by XSimMake can be fully simulated in a Viewlogic simulator. Figure 8-1 shows the XSimMake design flow.

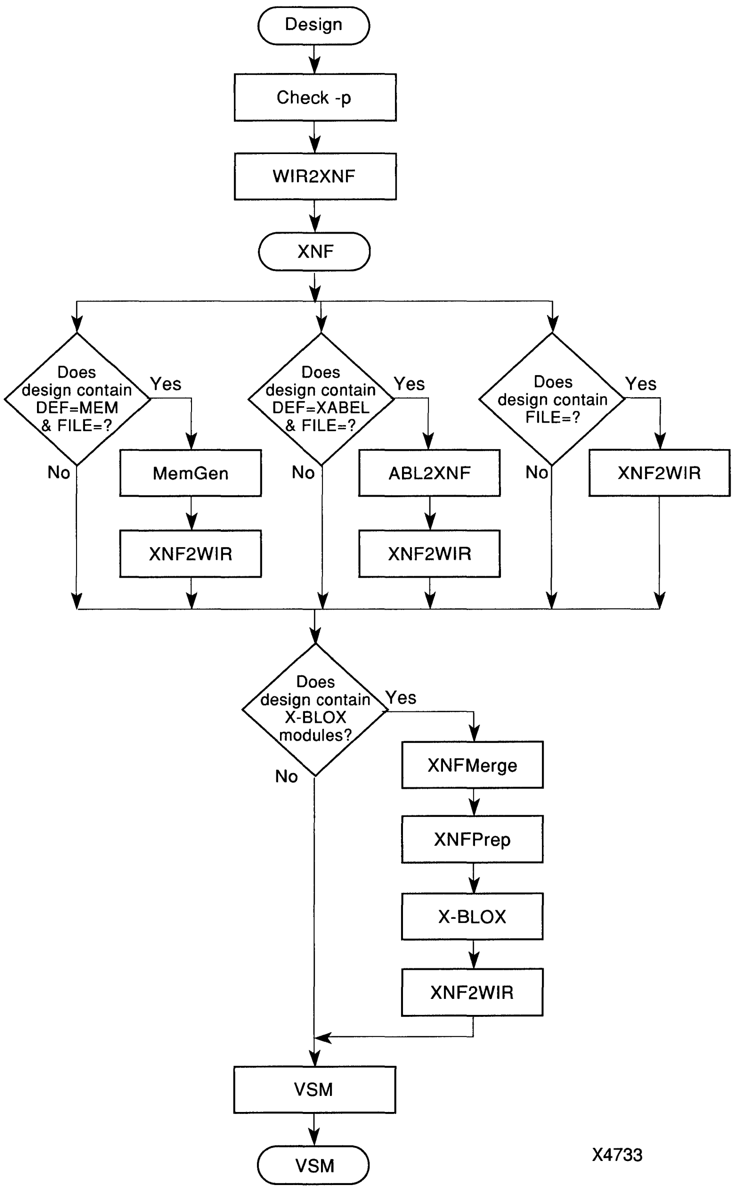

Design
Check -p
WIR2XNF
XNF
Does design contain DEF=MEM & FILE=?
Yes
No
MemGen
XNF2WIR
Does design contain DEF=XABEL & FILE=?
Yes
No
ABL2XNF
XNF2WIR
Does design contain FILE=?
Yes
No
XNF2WIR
Does design contain X-BLOX modules?
Yes
No
XNFMerge
XNFPrep
X-BLOX
XNF2WIR
VSM
VSM
X4733

Figure 8-1 XSimMake Design Flow for FPGA Functional Simulation

Designs Containing Special Components

XSimMake allows you to create simulation schematics for special components. Special components are hard macros, symbols with FILE attributes, X-BLOX components, Xilinx ABEL state machines, MemGen components, CLB and IOB primitives, and any other blocks without schematics. These components have no netlists available for the simulator to use and therefore require special preparation for simulation.

In order to simulate these special components on your schematic, you must run XSimMake. XSimMake, when used for functional simulation, reads in a schematic and outputs both a VSM file for functional simulation and new schematics located in a subdirectory of the current project directory. During simulation, the back-annotated simulation values on the special components are annotated to the new schematics, and the logic states on all the pins of the components as well as on all the nets and buses are displayed.

If your design contains special components, you *must* run XSimMake with the functional flow option selected before using XSimMake for timing simulation. The functional flow option creates files used by the timing simulation flow. You do not actually have to perform a functional simulation before simulating the timing; you just have to run XSimMake with the functional option selected.

The following sections briefly explain the steps that must be taken for each special symbol before you can run XSimMake.

Note: After running XSimMake and returning to your schematic editor, close your original schematic before accessing the simulation schematic so that the modified viewdraw.ini file generated by XSimMake will be re-read.

Designs Containing User-Created Hard Macros

You must convert user-created hard macros to relationally placed macros (RPMs) with the HM2RPM utility, described in the "HM2RPM" chapter of the *Development System Reference Guide.*

Your viewdraw.ini file must contain the SHM4000 library if your design includes pre-XACT 5.0 Xilinx-supplied hard macros. Be sure to place the SHM4000 library right after the primary library in the viewdraw.ini file. Warnings may occur if it is not placed right after the primary library.

Note: In general, add the SHM4000 library to your viewdraw.ini file for XC4000 designs. It contains symbols that prevent you from receiving warnings from XNF2WIR.

Designs Containing FILE Attributes

Designs containing any blocks with a netlist referenced by a FILE attribute, such as PALs, must be processed using the following sequence of commands before performing a functional simulation.

1. Make sure the newly created XNF file is in the current directory or placed in the xnf subdirectory. XSimMake only looks in those two locations to find the XNF file.
2. Place a symbol with the FILE=*filename* attribute on your schematic to reference the XNF file.
3. Save the schematic and close your design entry tool.
4. Invoke XSimMake, specifying the top-level schematic name as the file design name.
5. Open the new s*design* schematic, and load the s*design*.vsm simulation netlist.

Designs Containing X-BLOX Components

An X-BLOX schematic contains X-BLOX modules that are connected by buses. These buses are labeled, but the labels do not contain any information about the width of the bus. Bus width information is supplied by the BOUNDS attribute attached to a BUS_DEF, INPUTS, OUTPUTS, or BIDIR_IO module. Figure 8-2 shows a simple example of an X-BLOX schematic containing buses whose labels indicate the bus name but not the bus width.

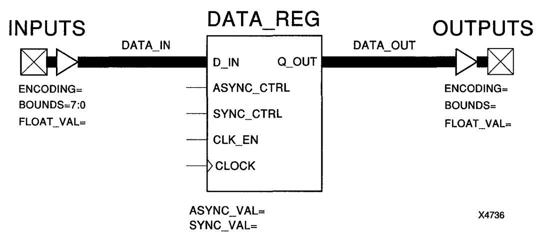


Figure 8-2 X-BLOX Bus Labels Before X-BLOX Processing

In this example, there are two buses: one labeled DATA_IN and the other labeled DATA_OUT. When the design is processed by X-BLOX, these buses are expanded into eight bits because of the BOUNDS attribute on the INPUTS module. When the VSM file is created for this design, these buses are referenced as DATA_IN[7:0] and DATA_OUT[7:0]. If you do not run XSimMake and try to view the simulation results on this schematic, you will see question marks for the simulation values on the buses because the bus name in the VSM file does not match the bus name on the schematic. XSimMake addresses this problem by creating a modified schematic with bus names that match those in the VSM file, as in Figure 8-3.

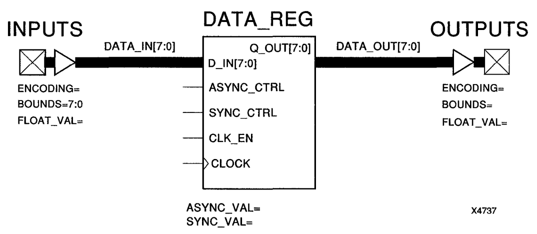


Figure 8-3 X-BLOX Bus Labels After X-BLOX Processing

The new schematic displays modified pin names on the X-BLOX modules and modified bus names to reflect the width specified by the BOUNDS attribute. Simulation values are now properly displayed on the schematic.

To simulate a design containing X-BLOX modules, follow these steps.

1. Instantiate the X-BLOX components in your schematic. Be sure to place the appropriate attributes on the components. See the *X-BLOX Reference/User Guide* for more information.
2. Save the schematic and close your design entry tool.
3. Invoke XSimMake, specifying the top-level schematic name as the file design name.

   To simulate X-BLOX modules on your schematic, you must run XSimMake. XSimMake creates the VSM file, but it also writes out new schematics because of the nature of X-BLOX pin and bus naming conventions. XSimMake creates a new schematic with the pins and buses expanded to the correct widths.
4. Open the new s*design* schematic, and load the s*design*.vsm simulation netlist.

Use a Viewlogic simulator to simulate the s*design* schematic. The simulator displays the back-annotated simulation values of the special components on your schematic and the logic states on the pins of the components as well as on the nets and buses.

Note: As noted previously, you *must* use XSimMake with the functional flow option selected before using XSimMake for timing simulation, because the functional flow creates files that are read by the timing simulation flow. You do not actually have to perform a functional simulation before simulating the timing; you just have to run XSimMake with the functional flow option selected.

Designs Containing Xilinx ABEL Components

Follow this sequence of commands before you generate a functional simulation netlist with XSimMake on FPGA designs containing Xilinx ABEL components.

1. Create the ABEL-HDL (ABL) file.
2. Convert the ABL file to an XNF file by using the Abl2xnf *filename* command. This command creates three files. The file with the .xsf extension is the symbol description file, the file with the .xnf extension is the Xilinx netlist file, and the file with the .xas extension is the simulation file.

3. Run SymGen with the –v option on the XSF file, or use the Symbol Generation utility in Windows.
4. Place the new symbol on your schematic. This symbol represents the ABEL-HDL file.
5. Save the schematic and close your design entry tool.
6. Run XSimMake, specifying the top-level schematic name as the design name.
7. Open the new s*design* schematic and load the s*design*.vsm simulation netlist.

Designs Containing MemGen Components

To perform functional simulation on a design containing MemGen components, follow this procedure.

1. Use MemGen to convert the MemGen MEM file to an XNF file. MemGen creates a symbol.
2. Edit the MEM file to place the initial values of the ROM.
3. Re-run MemGen to create the correct XNF file with the RAMs or ROMs.
4. Place the new MemGen-created component in your schematic.
5. Save the schematic and close your design entry tool.
6. Invoke XSimMake, specifying the top-level schematic name as the design name.

   In addition to the new schematic, s*design*, and s*design*.vsm, XSimMake also creates an s*design*.xmm file when it invokes XNF2WIR. It creates this file whenever a design contains ROMs, whether they are MemGen, X-BLOX, or library ROMs. This file contains Loadm statements that initialize the ROMs. You can use the XMM file directly before executing the command file, or you can concatenate it to the beginning of the command file.
7. Open the new s*design* schematic and load the s*design*.vsm simulation netlist.
8. After the VSM file has been loaded into the simulator, run the XMM file manually to initialize the ROMs in the simulation window or in a simulation command file.

Designs Containing CLB and IOB Primitives (XC2000, XC3000, and XC3100 Only)

If your XC2000, XC3000, or XC3100 design contains any CLB or IOB primitives, you must create an unrouted LCA file as the basis of your functional simulation netlist. Please refer to the "Translating Designs Containing IOB or CLB Primitives to VSM Files" section later in this chapter for instructions on this procedure.

Invoking XSimMake

To process an FPGA design, use the following syntax to run XSimMake.

Note: Because XSimMake appends an "s" to the design name, the name must be seven characters or fewer for PC users.

XSimMake -f viewlogic_fpga_func [–*options*] *design_name*

or you can use the shortened form of this syntax:

XSimMake -f vff [–*options*] *design_name*

Options can be any of the following options.

- –f

  The –f option specifies the type of simulation, or flow, that you want XSimMake to run. For Viewlogic, these types are the following:

  - VFF — FPGA functional simulation
  - VEF6 — EPLD functional simulation
  - VFT — FPGA timing simulation
  - VET — EPLD timing simulation for devices designed with software pre-dating the XACT*step* Version 6.0 software.
  - VET6 — EPLD timing simulation for devices designed with XACT*step* Version 6.0 software.

  Use the following syntax to select a flow:

  XSimMake -f {VFF|VEF6|VFT|VET|VET6} *design_name*

- –h

  The –h option offers help on the XSimMake syntax. For more specific help on each of the available flow options, type the –h option with the flow. The syntax is the following:

  `XSimMake -h {VFF|VEF6|VFT|VET|VET6}`

- –l

  The –l option lists the flows that XSimMake can run. These flows are listed under the –f option.

- –o

  The –o option specifies that the libraries created before the Unified Libraries will be used to process the design.

- –p

  The –p option specifies the part type for the design.

- –r

  The –r option forces the re-execution of all programs used by XSimMake. By default, XSimMake checks to see if these programs need to be re-run.

- –v

  This option enables you to view the progress of the programs executed by XSimMake on your computer screen.

Note: If this option is used, an xsimmake.out output file is not produced.

EPLD Designs

For EPLDs, XSimMake reads any behavioral modules you may have in your schematic and automatically creates a simulation netlist that represents their functionality. XSimMake also finds any INIT attributes and configures the simulation netlist to initialize your registers in response to the PRLD or MRESET pulse. Registered components with no INIT attribute will not be initialized during functional simulation and will therefore contain an initial value of X.

Invoking XSimMake

To process an EPLD design, use the following syntax to run XSimMake. You can only run XSimMake for functional simulation on EPLDs when using XACT*step* 6.0 software.

XSimMake -f viewlogic_epld_func6 [*–options*] *design_name*

or you can use the shortened form of this syntax:

XSimMake -f VEF6 [*–options*] *design_name*

Options are the same as those listed in the previous section, "FPGA Designs."

Using XMake and XEMake to Implement Designs

XMake is a Xilinx program that translates an FPGA schematic design into a logic cell array (LCA) file. Similarly, XEMake is a Xilinx XACT V5.*x* program that translates an EPLD design into a VMH formatted design database file. With XMake for FPGAs or XEMake for EPLDs, you can translate and implement your design automatically in one step.

An LCA file is the implemented version of an FPGA design, that is, a design that was placed and routed by the Xilinx core tools. Once you have generated a BIT file using MakeBits, you can download the BIT file to your target system using XChecker on workstations or DOS-based systems or the Hardware Debugger for Windows-based systems. Alternatively, you can create a PROM file using the MakePROM program on workstations or DOS-based systems or the PROM File Formatter for Windows-based systems.

A VMH file is an EPLD database file that is created by the Xilinx fitter. FITNET reads the design file in XNF format and fits it to an EPLD device. XEMake optionally creates a programming (PRG) file in Intel HEX format.

Before translating schematics, memory compilers, state machines, Boolean equations, or other design entry format files into an LCA file for FPGAs, you must first convert the design entry files into Xilinx netlist format (XNF) files. In EPLD designs, only the schematic information is converted to XNF format; equation files remain separate. An XNF file describes each logic primitive in a design, its associated pins, the connections between the primitives, and any parameters you have specified on the symbols. For information on the conversion process, see the "Creating a Xilinx Netlist Format (XNF) File" later in this chapter.

FPGA Designs

The XMake program automatically executes the translation programs needed to convert an FPGA design into a routed LCA file and optionally, a BIT file, as shown in Figure 8-2. Using the WIR2XNF and XNFMerge programs, it first converts the design file into a Xilinx netlist format (XNF) file to make the design compatible with the Xilinx core tools. Then it runs the appropriate design implementation tools in succession. Even designs that contain both schematic and non-schematic modules, such as memory modules and Boolean equations, are automatically translated. The program can process your design successfully given a WIR subdirectory file, which contains wirelist data, or a MAK file.

Note: You can make minor changes to your design and recompile it using the original design as a guide file for any of the FPGA families. For information on this capability, see the *Development System Reference Guide.*

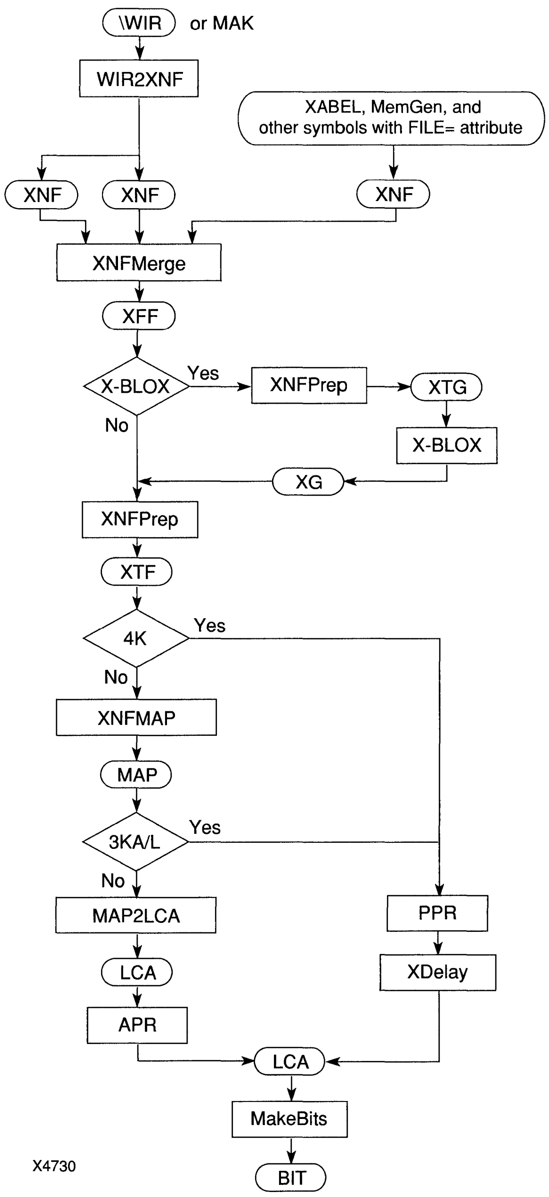


Figure 8-2 Implementing an FPGA Design with XMake

Invoking XMake

You can invoke XMake from the operating system command line by using the following syntax:

```
xmake [–options] toplevel
```

or

```
xmake [–options] toplevel.mak
```

Toplevel.mak is a MAK file, and *options* can be any of the options given in the "XMake" chapter of the *Development System Reference Guide.*

You can use the –p option to specify the intended part type. The –l option is required to process designs that reference the libraries created before the Unified Libraries. The –l option also ensures that ABL2XNF and MemGen produce an XNF file that is compatible with the old libraries. If you do not specify the –l option, XMake generates XNF files with symbols from the Unified Libraries; these files are incompatible with the old library symbols used in the rest of your design.

Some of XMake's options control how MAK files are created. These options are used by programs invoked automatically by XMake. Except for the –r option, these options are only valid when you specify a design file. They do not apply if you invoke XMake with a MAK file, since the MAK file already describes which programs and options to use. Refer to the *Development System Reference Guide* for detailed information on the XMake options.

XMake automatically invokes the appropriate translators to convert the top-level design into an LCA file.

When a design file is used as an input file, XMake saves the translation commands in a file called *toplevel*.mak, which you can then use as an input to subsequent runs of XMake, provided that you have not modified the hierarchical structure of your design.

Table 8-4 summarizes the programs that XMake runs automatically.

Table 8-4 XMake Design Processing

XMake performs the following steps.		
WIR2XNF	Translates the WIR subdirectory file into one or more XNF files.	All designs
XNFMerge	Merges all XNF files into one flat XNF format file (XFF file). XNF files generated by Xilinx ABEL and MemGen are also merged.	
If there are X-BLOX modules, XMake performs these additional steps:		
XNFPrep	Performs DRC checking and trims unused/disabled logic. Generates an XTG file, which is a trimmed XNF format file.	X-BLOX only
X-BLOX	Synthesizes the X-BLOX modules and generates an XNF format file (XG file).	
For all FPGA designs, XMake performs these steps:		
XNFPrep	Performs DRC checking and trims unused/disabled logic. Generates an XTF format file.	All designs
XNFMAP	Maps the XTF file logic into logic and I/O resources of the FPGA. Generates a netlist (MAP file) mapped into CLBs, IOBs, BUFTs, and clock buffers.	XC2000, XC3000, XC3000A/L, and XC3100A designs
MAP2LCA	Translates MAP file into an unrouted LCA file.	XC2000 and XC3000 designs
APR	Generates a placed and routed LCA file with delays.	
PPR	Generates a mapped, placed, and routed LCA file.	XC3000A/L, XC3100A, and XC4000 designs
XDelay option -dw	Writes delays into your LCA file.	
MakeBits	Processes the LCA file and generates a BIT file.	All designs

For additional information on XMake and design implementation, refer to the *Development System Reference Guide.*

EPLD Designs

The XEMake program automatically translates EPLD design schematics into XNF files when you use XACT V5.*x* software. XEMake automatically converts a design into a VMH file and optionally a HEX programming file using WIR2XNF, XNFMerge, PLUSASM, FITNET, and MakePRG in succession. The program can process your design successfully given a schematic or a MAK file.

The XEMake program automatically executes the translation programs needed to convert a design into VMH and HEX files, as shown in Figure 8-3. Even designs that contain both schematic and non-schematic modules, such as Boolean equations, can be automatically translated.

If you have existing schematics that were created with an earlier version of the XEPLD Viewlogic library, you can continue to use your earlier library to process your designs. However, you must create new XNF netlists from your schematics using XEMake or WIR2XNF. XEPLD no longer accepts EDIF formatted netlist files.

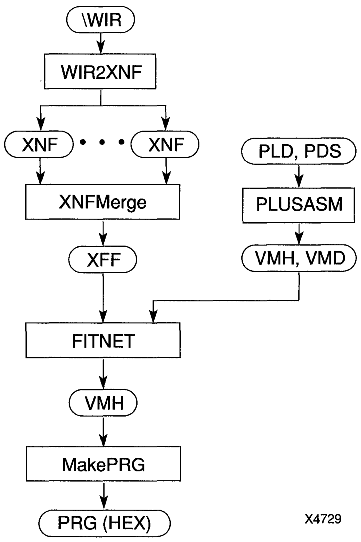


Figure 8-3 Implementing an EPLD Design with XEMake

Invoking XEMake

You can invoke XEMake from the operating system command line by using the following syntax.

```
xemake [–options] toplevel.1 [target.prg]
```

or

```
xemake [–options] toplevel.mak
```

Toplevel.mak is a MAK file, and *options* can be any of the options described in the" XEMake" section of the *XEPLD Reference Guide.* Use the –p option to specify the intended part type. The –s (signature) option specifies a signature string whenever you specify *target*.prg.

If you want to create an Intel HEX file, specify a target file name with a .prg extension.

XEMake automatically invokes the appropriate translators to convert the top-level design into a VMH file.

When a design file is used as an input file, XEMake saves the translation commands into a file called *toplevel*.mak, which you can then use as an input to subsequent runs of XEMake, provided that you have not modified the hierarchical structure of your design.

Table 8-5 summarizes the programs that XEMake runs automatically.

Table 8-5 XEMake Design Process

WIR2XNF	Translates the WIR subdirectory files into one or more XNF files.
XNFMerge	Merges all XNF files into one flat XNF format file (XFF file).
PLUSASM (optional)	Assembles equation files for all PLD symbols in the design into VMH files in the CLIB subdirectory.
FITNET	Generates a VMH file containing the implemented EPLD design.
MakePRG (optional)	Generates a programming list file (PRG) in HEX format.

Using XSimMake to Create Timing Simulation Data

Note: If your FPGA design contains special components such as X-BLOX modules or Xilinx ABEL components, you must first run XSimMake with the functional flow option selected before you can use XSimMake for timing simulation.

FPGA Designs

In functional simulation for FPGAs, XSimMake creates a simulation directory called s*design* ("s" concatenated with your design name) under the design project directory. It creates subdirectories in the simulation directory called sch, wir, and sym: in essence, it makes a user-created library. It edits the viewdraw.ini file in the design project directory, adding the new simulation directory to the library list. The alias for the new library is s*design*. This addition to the viewdraw.ini file allows you to access the simulation directory without leaving the project directory and without changing projects. XSimMake writes the simulation netlist file into the project directory, calling it s*design*.vsm.

When used for FPGA timing simulation, XSimMake reads in the routed LCA file and old functional VSM file, then outputs a new timing VSM file that includes all the timing information. XSimMake's timing flow does not create a schematic. You still use the schematic created during XSimMake's functional simulation run.

The XSimMake timing simulation flow for FPGAs is shown in Figure 8-4. Xilinx strongly recommends that you run XSimMake instead of translating manually, especially if your design contains special components such as X-BLOX, Xilinx ABEL, or MemGen components.

Use the s*design* schematic created during FPGA functional simulation. This schematic back-annotates the simulation values to the special components on your schematic so you can see the logic states on the pins of your components as well as on the nets and buses.

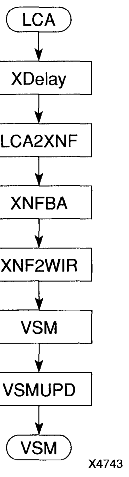


Figure 8-4 XSimMake Design Flow for FPGA Timing Simulation

Designs Containing Special Components

If your design contains special components, you *must* run XSimMake with the functional flow option selected before using XSimMake for timing simulation. The functional flow option creates files used by the timing simulation flow. You do not have to actually perform a functional simulation before simulating the timing; you just have to run XSimMake with the functional flow option selected.

See the "Using XSimMake to Create Functional Simulation Data" earlier in this chapter for more information on designs containing special components.

Invoking XSimMake

To process an FPGA design, use the following syntax to run XSimMake:

> **xsimmake –f viewlogic_fpga_timing** [*–options*] *design_name*

or you can use the shortened form of this syntax:

> **xsimmake –f vft** [*–options*] *design_name*

Options can be any of the options described in the "FPGA Designs" section under the "Using XSimMake to Create Functional Simulation Data" heading earlier in the chapter.

EPLD Designs

For EPLD timing simulation, XSimMake reads either the VMH data base file (V5.*x*) or the xsimtime.xnf file (V6.*x*) that represents the actual logic paths in the EPLD device with worst-case timing, as configured by your design. XSimMake generates a Viewlogic wirelist file named xsimmake.1 (V5.*x*) or xsimtime.1 (V6.*x*) in your wir directory. The simulation wirelist file is not named the same as your design so that it is never inadvertently read during the design entry of a subsequent iteration. XSimMake then runs VSM to translate the wirelist file into a *design*.vsm file for simulation.

Figure 8-5 displays the timing simulation flow for EPLDs.

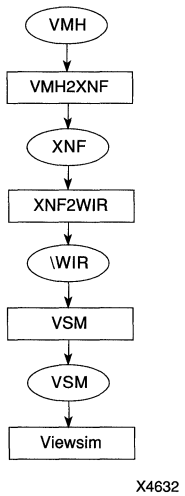


Figure 8-5 XSimMake Design Flow for EPLD Timing Simulation

Invoking XSimMake

To process an EPLD designed with XACT*step* Version 6.*x* software, use the following syntax to run XSimMake.

> **xsimmake -f viewlogic_epld_timing6** [*–options*] *design_name*

or you can use the shortened form of this syntax:

> **xsimmake -f vet6** [*–options*] *design_name*

To process an EPLD designed with XACT software pre-dating Version 6.0, use the following syntax to run XSimMake:

> **xsimmake -f viewlogic_epld_timing** [*–options*] *design_name*

or you can use the shortened form of this syntax:

> **xsimmake -f vet** [*–options*] *design_name*

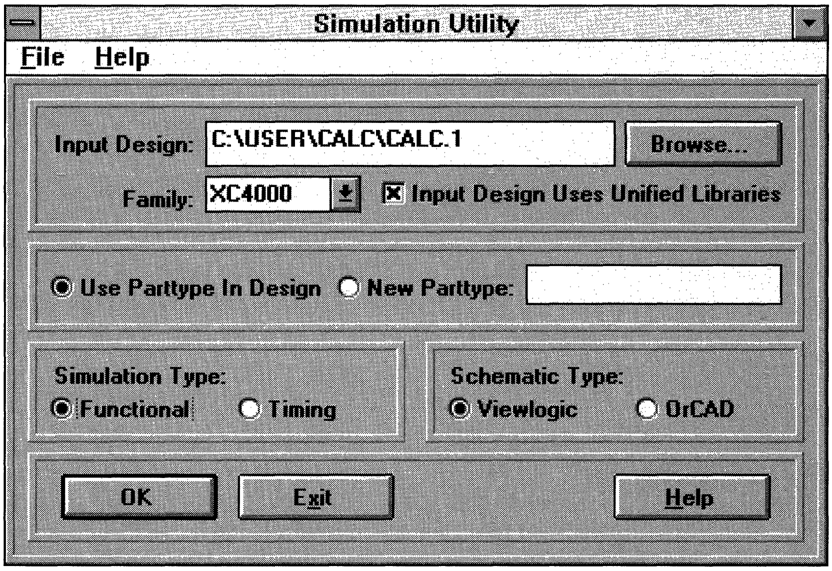
The options available in XSimMake are given in the "Using XSimMake to Create Functional Simulation Data" section. Normally, no options other than –f are required when running XSimMake.

Running XSimMake from Windows

You can run XSimMake manually from Windows without using PROflow. In Windows, XSimMake is called the Simulation Utility.

1. In the Program Manager XACT*step* program group, double-click on **Simulation Utility**.

   The dialog box shown in Figure 8-6 appears.



Input Design: C:\USER\CALC\CALC.1

Family: XC4000 [✓ Input Design Uses Unified Libraries]

[✓ Use Parttype In Design] New Parttype:

Simulation Type: [✓ Functional] Timing

Schematic Type: [✓ Viewlogic] OrCAD

Figure 8-6 Functional Simulation Utility Dialog Box

2. In the Input Design field, enter the name of the top-level design file.
3. In the Family field, select the device family of the design.
4. Select **`Use Parttype In Design`** if you want to use the part type that the design currently uses, or **`New Parttype`** to use a new part type. If you use a new part type, enter the name of the part in the box.
5. In the Simulation Type field, indicate whether you want to perform a functional or a timing simulation.
6. In the Schematic Type field, click on **`Viewlogic`** if it is not already selected; it is selected by default.
7. Click on **`OK`**.

Individual Programs

The remainder of this chapter describes the individual programs that XSimMake runs automatically.

Creating a Xilinx Netlist Format (XNF) File

You must translate your design into an XNF file before you can create an implemented design file or a simulation file. The WIR2XNF and XNFMerge programs are used sequentially to create a flattened XNF format (XFF) file from your Viewlogic WIR subdirectory files, as shown in Figure 8-7.

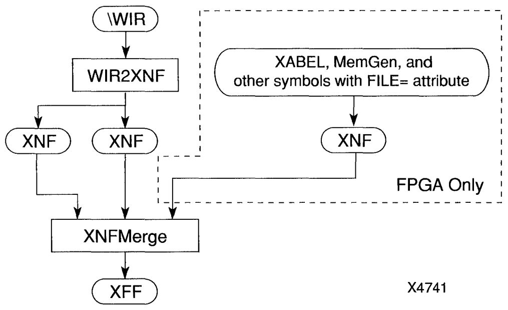


Figure 8-7 Creating XNF Files

WIR2XNF uses the WIR subdirectory files from Viewlogic as input files and converts them into the corresponding XNF files.

XNFMerge merges all XNF files for the design into a top-level flattened XNF format (XFF) file. XNF files for blocks without schematics, such as files created by Xilinx ABEL or MemGen, are also merged into the XFF file.

WIR2XNF

The WIR2XNF program converts the Viewlogic wirelist (WIR subdirectory) files into Xilinx netlist format (XNF). The WIR2XNF program traverses through your design hierarchy, creating an XNF file for each level of hierarchy and for each Xilinx macro.

Syntax

To execute the WIR2XNF program from the command line, you must be in the directory above the wir subdirectory and use the following syntax.

wir2xnf [*options*] *design* [*new_name*[**.xnf**]]

Input File

WIR2XNF accepts as input the *design*.1 netlist file located in the wir subdirectory.

Output File

WIR2XNF creates the following files:

- XNF files with the same name as the input schematic and lower-level schematics
- A CRS file containing aliases for certain signals. For example, XNF2WIR changes signals named ~RD and ~IRD in the Viewlogic file to -RD and -IRD, respectively, in the XNF file, and these aliases are stored in the CRS file.

Options

The WIR2XNF options are listed and described in the "Program Options" appendix.

Warning and Error Messages

The WIR2XNF warning and error messages are given in the "Warning and Error Messages" appendix.

Rule Checking

All Xilinx library components have PINTYPE attributes assigned to the pins. WIR2XNF uses these PINTYPE attributes to perform some basic design rule checks to ensure that there are no schematic errors in the design.

XNFMerge

You must merge the XNF files created by WIR2XNF together into a single flattened XNF file using the XNFMerge program.

XNFMerge merges all of the XNF files created by WIR2XNF, as well as the XNF files for blocks without schematics, such as Xilinx ABEL or MemGen blocks. The output of XNFMerge is a single flat XNF format file with an .xff extension.

Syntax

To execute the XNFMerge program from the command line, use the following syntax:

xnfmerge [*options*] *input_file*[**.xnf**] [*output_file*[**.xff**]]

Note: When you run XNFMerge on the command line and specify a part number, the part number cannot begin with "XC." It must begin with 2, 3, 4, 5, or 7. Here are two examples of part numbers incorrectly specified:

```
xnfmerge -p xc7236-30pc44

xnfmerge -p xc4005pq160-5
```

The correct way to enter these part numbers is like this:

```
xnfmerge -p 7236-30pc44

xnfmerge -p 4005pq160-5
```

Input Files

The files input to XNFMerge are described in the *Development System Reference Guide.*

Output Files

The files that XNFMerge produces are described in the *Development System Reference Guide.*

Options

The options available to modify the operation of XNFMerge are described in the *Development System Reference Guide.*

Creating a Functional Simulation (VSM) File

This section explains how to execute the individual programs necessary to translate your XNF file into a functional simulation network (VSM file).

EPLD Designs

For EPLD designs using XACT V5.*x* that contain no PLD symbols or custom primitives defined using equation files, use only the VSM program. Follow the instructions given in the "Creating VSM Files for Designs Containing No Special Components" section of this chapter. Designs containing custom PLD symbols or custom primitive symbols cannot be functionally simulated using XACT V5.*x*.

FPGA Designs

This section discusses the functional simulation flows used for FPGA designs.

A file used for functional simulation does not require any delay information, since the functional netlist file is only used to verify the logic in your design. The Xilinx XNF2WIR and VSM programs translate the XNF format file into a functional netlist.

To create a functional simulation file in the Viewlogic design flow, use one of the following files.

- Use the XFF file created by running XNFMerge on the top-level XNF file if your design contains Xilinx ABEL blocks.
- Use the XG file created by running X-BLOX on the XTG file if your design contains X-BLOX modules.
- Use the XTF file created by running XNFPrep on the XFF or XG file if your design contains CLB or IOB primitives.
- If the design has already been processed using XMake, use the placed and routed LCA file. If XMake is not used, this file is created by either APR for XC2000 and XC3000 designs or PPR for XC3000A/L, XC3100A, XC4000, and XC5200 designs.

Note: If your design contains no special components, such as X-BLOX modules or Xilinx ABEL blocks, refer to the "Creating VSM Files for Designs Containing No Special Components" section.

Before running the programs that generate the actual simulation file, you must ensure that all XNF files are merged into one flat XNF format (XFF) file with XNFMerge, that all X-BLOX modules are processed, and that all XC2000, XC3000, XC3000A/L, and XC3100A designs containing IOB and CLB primitives are processed with XNFPrep.

Figure 8-8 illustrates the different types of XNF format files (XFF, XG, and XNF) that are created, depending on the type of FPGA modules used in the design: logical primitives only, X-BLOX modules, or IOB and CLB primitives.

- Area 1 shows the flow used for designs containing only logical primitives and Xilinx macros.
- Area 2 shows the additional steps necessary to process X-BLOX modules.
- Area 3 shows the additional steps necessary to process designs that include IOB and CLB primitives.

Use the final XNF format file to run the necessary programs to create a VSM file.

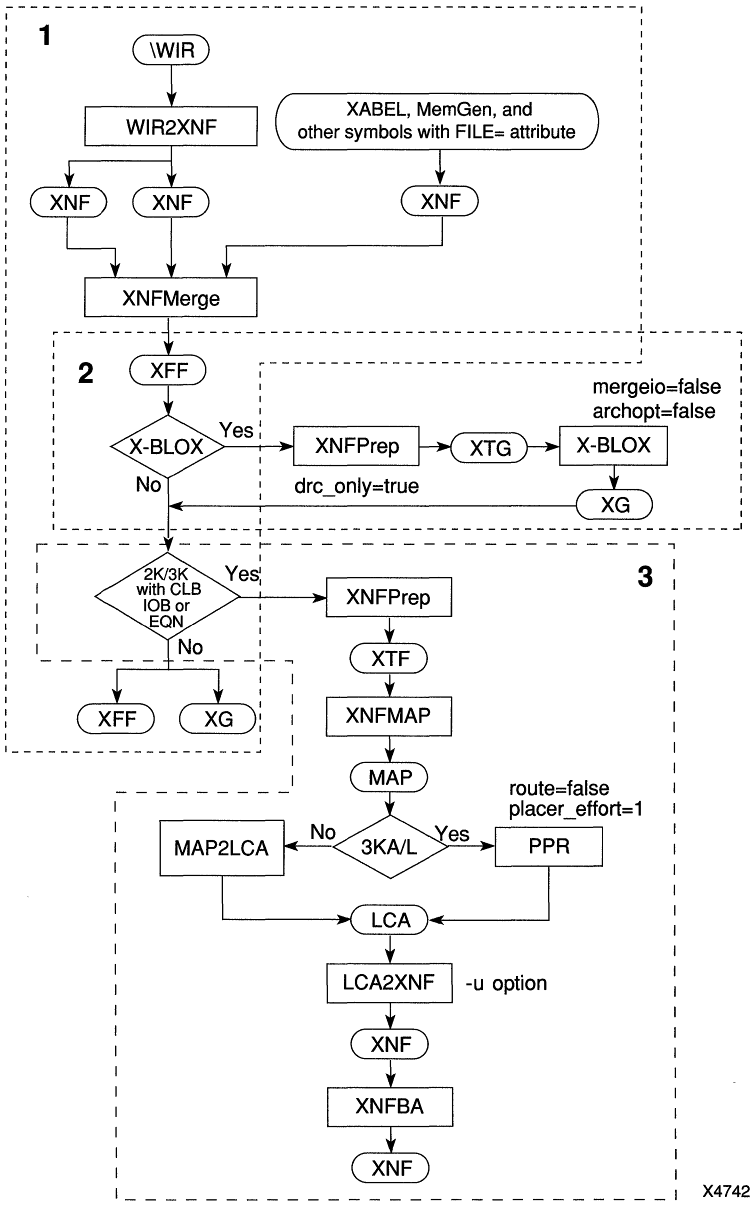


Figure 8-8 Creating XNF Files for FPGA Functional Simulation

Creating VSM Files for Designs Containing No Special Components

If your EPLD or FPGA design contains no special components, such as custom primitive symbols for EPLDs or X-BLOX modules for FPGAs, which have no underlying schematics, you can simulate the design without having to run WIR2XNF or any other Xilinx translation tool. All the necessary simulation models are already in your schematic. You can create your VSM file directly from the command line.

To create a VSM file directly, type **vsm** *design*↵.

When VSM is complete, you see the following message:

```
0 errors and 0 warnings. Press any key to con-
tinue.
```

Translating XFF Files to VSM Files

Follow these steps to convert a file created with WIR2XNF and XNFMerge into a VSM file for FPGA functional simulation.

1. Execute the XNF2WIR program from the command line using the following syntax:

   xnf2wir *filename*.**xff** *new_filename*

   XNF2WIR converts the Xilinx netlist back to a Viewlogic wirelist file. You must specify a new file name as the output so that you do not overwrite your existing WIR files. When you specify a different output file name for XNF2WIR, it writes the file into the current directory instead of the wir subdirectory. Before proceeding to the next step, you must move the new file from the current directory to the wir subdirectory by entering **wir**\new_filename. For example, to translate input.xnf to an output.1 WIR file, type **xnf2wir input.xnf wir\output.1** .

2. Execute the VSM program from the command line using the following syntax:

   vsm *new_filename* [**.1**]

   VSM converts the WIR subdirectory files into a simulation wirelist (VSM) file.

3. Execute the VSMUPD program from the command line with the following syntax:

   vsmupd *filename* [**.vsm**] *new_filename* [**.vsm**]

   where *filename* is the unrouted VSM file, and *new_filename* is the routed VSM file. These files are created by running the VSM program on the routed and unrouted WIR files. If the program does not find the unrouted VSM file, it automatically creates one from the unrouted WIR file.

   Use the *new_filename*.vsm file in simulation.

When your design is translated to Xilinx netlist format, some of the nets have their hierarchical references removed. VSMUPD adds net equivalences to the VSM file for these removed references, which facilitates a more nearly complete simulation.

If you have either X-BLOX modules or IOB and CLB primitives in your design, proceed to the next section.

Translating XG Files to VSM Files

If you are designing an XC3000A/L, XC3100A, XC4000, or XC5200 device and your design contains X-BLOX modules, you must run XNFPrep to write an XTG file before running the X-BLOX software.

1. Enter the following from the command line:

   xnfprep *filename* [**.xff**]

   xblox *filename* [**.xtg**] **mergeio=false archopt=false**

   The Mergeio and the Archopt options must be set to False to prevent the program from optimizing.

2. Execute the XNF2WIR program from the command line using the following syntax:

   `xnf2wir` *filename*`.xg` *new_filename*

   XNF2WIR converts the Xilinx netlist back to a Viewlogic wirelist file. You must specify a new file name as the output so that you do not overwrite your existing WIR files. When you specify a different output file name for XNF2WIR, it writes the file into the current directory instead of the wir subdirectory. Before proceeding to the next step, you must move the new file from the current directory to the wir subdirectory by entering **`wir`**\new_filename. For example, to translate input.xnf to an output.1 WIR file, type **`xnf2wir input.xnf wir\output.1`** .

3. Execute the VSM program from the command line using the following syntax:

   `vsm` *new_filename* [**`.1`**]

   VSM converts the WIR files into a simulation wirelist (VSM) file.

4. Execute the VSMUPD program from the command line with the following syntax:

   `vsmupd` *filename* [**`.vsm`**] *new_filename* [**`.vsm`**]

   Filename is the unrouted VSM file, and *new_filename* is the routed VSM file. These files are created by running the VSM program on the routed and unrouted WIR files. If the program does not find the unrouted VSM file, it automatically creates one from the unrouted WIR file.

   Use the *new_filename*.vsm file in simulation.

When your design is translated to Xilinx netlist format, some of the nets have their hierarchical references removed. VSMUPD adds net equivalences to the VSM file for these removed references, which facilitates a more nearly complete simulation.

Translating Designs Containing IOB or CLB Primitives to VSM Files

If your XC2000, XC3000, or XC3100 design contains IOB or CLB primitives, you must run XNFPrep to generate an XTF file and XNFMAP to map the XTF file into the logic and I/O resources of the FPGA.

1. Enter the following from the command line:

   xmake -n *design*

   For XC2000, XC3000, and XC3000A designs, XMake creates a MAP file.

2. For XC2000 or XC3000 designs, run MAP2LCA to generate an unrouted LCA file:

   map2lca *filename* [**.map**]

   For XC3000A/L or XC3100A designs, run PPR with the Route=false option and the Placer_effort=1 option. Refer to the "PPR" chapter of the *Development System Reference Guide* for more details.

3. See the following section, "Translating LCA Files to VSM Files," for instructions on translating the unrouted LCA file to a VSM file.

Translating LCA Files to VSM Files

Follow these steps to convert an LCA file into a functional simulation netlist (VSM) file.

1. Execute LCA2XNF with the –u option from the command line:

   lca2xnf -u *filename* [**.lca**] *new_filename* [**.xnf**]

   The –u option selects unit-delay translation. It is important to specify a new output file name for LCA2XNF to avoid confusion during the back-annotation process; it is required if XNFBA is to be used.

   The FPGA file contains signal names changed by XNFMAP for XC2000, XC3000, XC3000A/L, and XC3100A designs, or by PPR for XC4000 and XC5200 designs. Follow step 2 to restore any signal names that changed during design processing.

2. Execute XNFBA from the command line as shown following:

   `xnfba` *filename*`.`**`xtf`** *new_filename* [**`.xnf`**]

   XNFBA requires two input files: the unrouted netlist (XFF or XG file) and the routed XNF file created by LCA2XNF. XNFBA restores net names to the XNF file. By default, XNFBA creates an output file called xnfba.xnf; however, it is recommended that you run XNFBA with the –o option and use the same file name as *new_filename* to keep file names consistent throughout the back-annotation process.

3. Execute the XNF2WIR program from the command line using the following syntax:

   `xnf2wir` *new_filename* [**`.xnf`**]

   XNF2WIR converts the Xilinx netlist back to a Viewlogic wirelist file.

4. Execute the VSM program from the command line using the following syntax:

   `vsm` *new_filename* [**`.1`**]

   VSM converts the WIR subdirectory files into a simulation wirelist (VSM) file.

5. Execute the VSMUPD program from the command line with the following syntax:

   `vsmupd` *filename* [**`.vsm`**] *new_filename* [**`.vsm`**]

   where *filename* is the unrouted VSM file, and *new_filename* is the routed VSM file. These files are created by running the VSM program on the routed and unrouted WIR files. If the program does not find the unrouted VSM file, it automatically creates one from the unrouted WIR file.

   Use the *new_filename*.vsm file in simulation.

When your design is translated to Xilinx netlist format, some of the nets have their hierarchical references removed. VSMUPD adds net equivalences to the VSM file for these removed references, which facilitates a more nearly complete simulation.

Creating an Implemented Design File

This section summarizes how to generate implemented FPGA and EPLD designs.

FPGA Designs

To translate the XNF file (XFF, XG, or XTF) of an FPGA design into an implemented LCA file, use the Xilinx implementation tools. For more information, refer to the "Using XMake and XEMake to Implement Designs" section of this chapter and to the *Development System Reference Guide*. You must implement your design before you can create a timing simulation file.

EPLD Designs

Run FITNET on your schematic design to implement your EPLD design using XACT V5.*x*. If you have PLD components or custom primitives in your design, you must assemble each equation file using PLUSASM before running FITNET. Implementing your design is a requirement before generating a timing simulation file. To generate a programming (PRG) file in Intel HEX format, run MakePRG. Refer to the *XEPLD Reference Guide* for details.

Creating a Timing Simulation (VSM) File

A file used for timing simulation must contain delay information, which is generated when the design is placed and routed.

FPGA Designs

This section details the procedure used to generate a timing simulation file from your placed and routed FPGA design. Figure 8-9 shows the translators involved in generating this netlist.

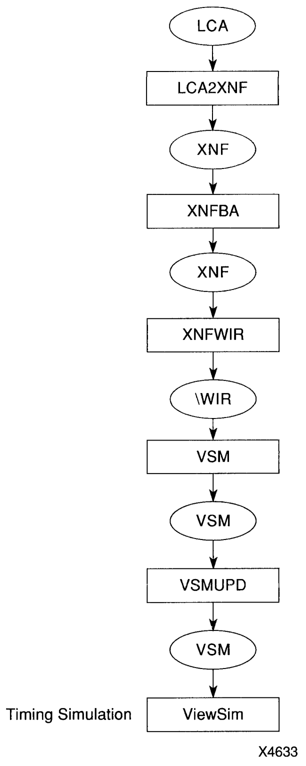


Figure 8-9 FPGA Timing Simulation Translation

If you created the LCA file without using XMake for an XC3000A/L, XC3100A, XC4000, or XC5200 design, you must run XDelay with the –dw option enabled on your LCA file; this option adds routing delay information to the LCA file. Then perform the following steps on the LCA file created by XDelay.

1. Execute LCA2XNF from the command line:

   `lca2xnf` *filename* [**`.lca`**] *new_filename* [**`.xnf`**]

   It is important to specify a new output file name for LCA2XNF to avoid confusion during the back-annotation process; it is required if XNFBA is to be used.

The LCA file contains signal names changed by XNFMAP for XC2000, XC3000, XC3000A/L, and XC3100A designs, or by PPR for XC4000 and XC5200 designs. Follow step 2 to restore any signal names that changed during design processing.

2. Execute XNFBA from the command line as shown here:

   xnfba *filename* [**.xg** | **.xff**] *new_filename* [**.xnf**]

   XNFBA requires two input files: the unrouted netlist (XFF or XG) and the routed XNF file created by LCA2XNF. XNFBA restores net names to the XNF file. By default, XNFBA creates an output file called xnfba.xnf; however, it is recommended that you run XNFBA with the –o option and use the same file name as *new_filename* to keep file names consistent throughout the back-annotation process.

3. Execute the XNF2WIR program from the command line using the following syntax.

   xnf2wir *new_filename* [**.xnf**]

   XNF2WIR converts the Xilinx netlist back to a Viewlogic wirelist file.

4. Execute the VSM program from the command line using the following syntax:

   vsm *new_filename* [**.1**]

   VSM converts the WIR subdirectory files into a simulation wirelist (VSM) file.

5. Execute the VSMUPD program from the command line with the following syntax:

   vsmupd *filename* [**.vsm**] *new_filename* [**.vsm**]

   where *filename* is the unrouted VSM file, and *new_filename* is the routed VSM file. These files are created by running the VSM program on the routed and unrouted WIR files. If the program does not find the unrouted VSM file, it automatically creates one from the unrouted WIR file.

When your design is translated to Xilinx netlist format, some of the nets have their hierarchical references removed. VSMUPD adds net equivalences to the VSM file for these removed references, which facilitates a more nearly complete simulation.

EPLD Designs

This section outlines the steps used to create a timing simulation file for EPLD designs using XACT V5.*x*.

To perform timing simulation, you must first integrate your design using XEMake, FITNET, or FITEQN for behavioral designs. The implemented design database file resulting from any of these procedures has a .vmh extension for all EPLD part types except XC7272, which uses a .vmd file extension. Then follow the procedure given in this section on the VMH or VMD file.

Figure 8-10 shows the steps required to generate a timing simulation netlist for EPLD designs.

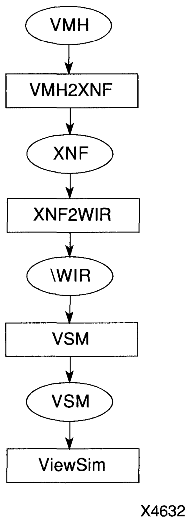


Figure 8-10 EPLD Timing Simulation Translation

To generate a timing simulation netlist file, follow these steps.

1. Run VMH2XNF from the command line to generate a timing simulation netlist file, as shown here:

   vmh2xnf *filename* **-o** *new_filename*

2. Execute the XNF2WIR program from the command line using the following syntax.

xnf2wir *new_filename* [**.xnf**]

XNF2WIR converts the Xilinx netlist file back to a Viewlogic wirelist file.

3. Execute the VSM program from the command line using the following syntax:

   vsm *new_filename*

VSM converts the WIR subdirectory files into a simulation wirelist (VSM) file. For board-level simulation, you can create a symbol for the EPLD device named *new_filename* and run VSM on the board-level schematic.

Translation Programs for Simulation

This section includes information about the programs used for Viewlogic simulation preparation. Included are sections on the XNF2WIR, VSM, VSMUPD, VMH2XNF, and ViewGen programs. For information about the XNFBA and LCA2XNF programs, which are also used in the translation process, refer to the *Development System Reference Guide.* This section applies to both functional and timing simulation programs.

XNF2WIR

XNF2WIR takes an XNF file and translates it to a WIR subdirectory file. If the XNF file was produced from a routed LCA file, all the delay information is passed to the WIR subdirectory file. With this information, you can perform timing simulation.

The XNF2WIR program outputs a WIR subdirectory file that is automatically placed in the wir subdirectory of the current directory.

Invoking XNF2WIR

The syntax for the XNF2WIR program, when invoked from the operating system command line, is the following:

xnf2wir [*options*] *input_filename* [**.xnf**] [*output_pathname*]

Input File

XNF2WIR uses the XNF file *input_filename*.xnf in the current directory as its input.

Output File

XNF2WIR creates a WIR file, *output_pathname*.1. By default, this file is given the same name as the input file with a .1 extension and placed in the wir subdirectory of the current directory. If an output file name is specified, the path must also be specified; otherwise, XNF2WIR places the file in the current directory. For example, to translate input.xnf to an output.1 WIR file, type **`xnf2wir input.xnf wir\output.1`** .

Note: To prevent any confusion between the files and ensure accurate simulation results, you should not allow the names of the back-annotated files to match those of the original files. If you have a multiple-page schematic, XNF2WIR still creates a single WIR file that contains the whole design. If the other WIR files with the same design name still exist at the time of wirelisting, VSM reads and processes all of them. This processing causes erroneous simulation results due to the duplications of some parts of the circuit. It is recommended that you change your design name during the LCA2XNF or VMH2XNF step of the timing simulation flow.

VSM

VSM is Viewlogic's wirelister, which creates a wirelist (VSM) file from a wire (WIR) file. The wirelist file can be simulated with a Viewlogic simulator.

Invoking VSM

To invoke VSM from the operating system command line, follow these steps:

1. Invoke VSM from the project directory, which is one level above the wir directory. This directory should also contain the viewdraw.ini file.
2. Type in the following command:

   `vsm` [*WIR_filename*] [*options*]

WIR_filename is the name of the input WIR file, also called a project, and *options* can be any of the options listed in the "VSM Options" section of the "Program Options" appendix.

Input Files

The input to VSM is a WIR file.

Output Files

The output of VSM is a wirelist VSM file.

Example

Following is an example of the syntax required to run VSM.

```
vsm alu -falu_std -h -daludelay.dtb
```

In this example, "alu" is the back-annotated WIR input file; "alu_std" is the name of the output wirelist file; –h generates full hierarchical net name equivalents in the wirelist file; and the back-annotation file name is "aludelay.dtb."

VSMUPD

A convenient way to simulate timing is to simulate on the original schematic. If your design is hierarchical, it is often necessary to push down into the hierarchy to verify simulation values. For FPGA designs, VSMUPD updates the VSM file to allow the back-annotation of signals that pass through hierarchy. It adds net equivalents from the original VSM file in the routed VSM file so simulation values do not appear as question marks in the simulation. It allows you to see the simulation values of this hierarchy in the context of the surrounding schematic rather than forcing you to push up out of the symbol to view the values, so you can easily verify that the simulator is simulating correctly.

As its inputs, VSMUPD uses an unrouted and a routed VSM file and a routed XNF file that has been optionally processed by XNFBA. The XNFBA program creates an xnfba.xnf file that is similar to the unrouted XNF file, except that the output xnfba.xnf file contains delay information. After converting the XNF file to a WIR subdirectory file and then to a VSM file, run VSMUPD on the VSM files to update the routed VSM file with information from the unrouted VSM file.

Running VSMUPD does not alter the delay or timing information. It only adds net-equivalent statements to the VSM file, enabling a Viewlogic simulator to back-annotate more net values to the original schematic.

Invoking VSMUPD

The syntax of the VSMUPD program, when invoked from the operating system command line, is the following:

> **vsmupd** [*options*] *unrouted_filename* [**.vsm**]
> *routed_filename* [**.vsm**]

To back-annotate your original schematic fully, you should run XNFBA on the routed FPGA XNF file before running VSMUPD.

Input Files

The input for VSMUPD is both the unrouted and routed VSM files, which are created by running the VSM program on the routed and unrouted WIR files.

If the program does not find the unrouted VSM file, it automatically creates one from the unrouted WIR file.

VSMUPD also uses the routed XNF file as input. The file name is assumed to be the same as the routed VSM file unless specified otherwise with the –x option.

Output File

VSMUPD outputs a *routed_filename*.vsm file. Unless the –o option is used, VSMUPD overwrites the routed input VSM file. The output of VSMUPD is used as input to a Viewlogic simulator.

VMH2XNF

VMH2XNF creates an XNF file with timing parameters for use in EPLD timing simulation when you use XACT V5.*x*. The input file can be a VMH or VMD (from XC7272) file.

Invoking VMH2XNF

The syntax to invoke VMH2XNF from the operating system command line follows:

vmh2xnf *design_name* **[-l] [-o]**

Input File

VMH2XNF accepts as input a VMH file output by XEMake, FITNET, or FITEQN for behavioral designs for all EPLD part types except XC7272, which outputs a VMD file.

Output File

The XNF file produced by VMH2XNF is written to the current directory by default, not the xnf subdirectory, so it will not overwrite the XNF file produced by WIR2XNF unless you change directory and output file options.

This XNF file is different from the XNF file produced by WIR2XNF, which is based on the schematic and used for design capture. The XNF file produced by VMH2XNF contains only simulation primitives with actual timing parameters. It contains an image of the EPLD device constructed from the bitmap in the VMH file. It does not consist of the library symbols used to capture your design and cannot be used as an input file to the software. The names of signals connected to device pins and component outputs are preserved except where component outputs are optimized.

ViewGen

This section provides information about ViewGen, Viewlogic's facility for generating a schematic from a single WIR file for FPGAs. Included in this section are an introduction to ViewGen and instructions for using ViewGen.

Purpose

ViewGen can automatically create schematics from a single WIR file (netlist) for FPGA designs. Options are available to control the results for parameters such as sheet size, page splitting, and labeling. You can specify these parameters with command line options or, in some cases, you can obtain them from the viewdraw.ini file.

Preliminary Requirements

To run ViewGen, a single WIR file must be available for input. Also, the external nets to be brought out to input or output connectors must be defined. Usually input, output, or bidirectional pins are included in the netlist to define external nets. In addition, external nets are connected to schematic pin components (I/O connectors) in the schematic.

Schematic Format

The schematic created by ViewGen can be spread across several sheets or placed within a single variable-sized sheet.

Within each sheet, input pins and off-page connectors are on the left, followed by logic gates connected in a left-to-right flow, followed by the output pins and more off-page connectors.

For tall and thin schematics, for example, a PAL with two levels of logic, you can place several columns or slices side by side on a single sheet.

An automatically generated border and title block are placed on the sheet; alternatively, you can specify a border and title block.

Connectivity

No loss of information occurs when creating schematics with ViewGen. The WIR files created from the generated schematics have the same connectivity as the original WIR file.

Placement and Visibility

If labels and attributes are defined to be visible, ViewGen displays them on components in the schematic. In addition, a placeholder for the label or attribute appears in the symbol definition. ViewGen places other component attributes at the component origin and makes them invisible.

ViewGen places labels for nets at an endpoint, either at a connection or a dangling endpoint (square joint). It makes visible labels at dangling endpoints, which represent off-page connectors; it makes all other labels on nets invisible. If you do not see label names, use the command from the menu or enter **`lvis`** at the command line.

ViewGen renders attributes invisible and places them at an end point or centers them on a net segment. It makes nets defined as global in the input netlist or defined as global with the NETNAME option in the viewdraw.ini file global.

Invoking ViewGen

You can invoke ViewGen only from the operating system command line.

The syntax of the ViewGen program is the following:

`viewgen` *WIR_file* [**`.1`**] *outfile options . . .*

The command line parameters are detailed in the following sections.

Input File

As an input file, ViewGen uses the WIR file from which the schematic is generated. You can specify a search path; if not, the current directory and then the wir subdirectory are searched. If the file is still not found, the search order in the viewdraw.ini file is searched. A file extension is optional; however, if it is omitted, a file extension of .1 is assumed.

Output File

The output file name is optional. If not specified, it is assumed to be the same as the input file name. If the output file ends with a "/" or "\," it is assumed to be a directory path, and the default file name is appended. If the output file is omitted, the default directory is the parallel sch subdirectory relative to the input file directory.

If multiple schematic pages are generated, the file extension in the output file is ignored and is replaced by the page number.

All files with the same name in the output directory are deleted before new files are written.

The default file extension for a single page schematic is .1.

Sheet Symbols

You can place a special annotation symbol in the generated schematic to serve as a border symbol with a page frame and title block. Unattached attributes placed within the sheet symbol identify the location and type for text typically placed in the title block.

When the schematic sheet is generated, the sheet symbol is instantiated in the appropriate position, and a combination of built-in and user-supplied attribute values are attached to the component (instance) of the symbol. The sheet symbol is defined with the –Sheetsym command option, and a separate symbol can be used for each sheet size.

Contents

The sheet symbol should include the lines that form the sheet frame, as well as a title block with attributes where values are to be placed. Additionally, the symbol should be of type Annotate, and it should not have any pins.

You must define the interior of the page where the drawing is placed with two unattached attributes, LOWER_LEFT and UPPER_RIGHT. The rectangle defined by the coordinates of these attributes is the region in which the drawing is centered. To avoid overlap with the drawing data, place the title block outside this rectangle.

Attributes

When the sheet symbol is placed in the generated schematic drawing, a built-in set of attributes is attached to this instance, and their values are modified to provide information about the drawing.

The location, size, and orientation for these attributes are defined by placing an unattached attribute with a matching name at the appropriate location on the symbol.

The attributes attached are a combination of the -Attr option and the built-in attributes listed in Table 8-6.

Table 8-6 Built-in Attributes

Attribute	Description
DESIGN_NAME	Name of the design as defined in the WIR file
USER_NAME	User login name
SHEET	Sheet number
SHEETOF	Highest sheet number generated (number of sheets)
WIR_FILE_NAME	WIR file name without path or extension
WIR_FILE_PATH	Full WIR file name
WIR_FILE_DATE	Creation (modification) date of the WIR file
WIR_FILE_TIME	Creation time of the WIR file
SCH_FILE_NAME	SCH file name without path or extension
SCH_FILE_PATH	Full SCH file name
SCH_FILE_DATE	Creation date of the SCH file
SCH_FILE_TIME	Creation time of the SCH file
SCH_PGM_NAME	Name of the program (ViewGen)
SCH_PGM_VER	ViewGen program version

Options

Two sheet symbol options are available.

- **–attr** *name=value name=value*

The –Attr option specifies one or more attribute names and values that are to be associated with the instance of the border symbol. This option also supplies values to attributes that you do not build in but place on the border symbol. In addition, some of the built-in attribute values, such as *design_name*, *user_name*, and WIR*, can be overwritten with this option. Multiple –Attr options are appended.

- **`-timeformat`** *n*

This option specifies a code that controls the conversion time and date strings used for the date and time built-in attributes. The values for *n* are defined in Table 8-7. The default code is 3.

Table 8-7 Codes for Specifying Time and Date

Code	String	Example
0	MM/DD/YY HH:MM	7/15/89 15:45
1	DD/MM/YY HH:MM	15/07/89 15:45
2	Month DD, YYYY HH:MM	Jul 15, 1989 15:45
3	DDMonthYY HH:MM	15Jul89 15:45
4-7	Same as 0-3, but time specified as HH:MM:SS	

Multiple WIR Files

ViewGen only reads one WIR file, but you can combine multiple WIR files into one WIR file by using the VSM –w option, which generates a WIR file-formatted netlist. This option combined with the –l (Level) option allows you to flatten the design to any specified level. ViewGen can now use the file to generate schematics.

Viewlogic Interface Guide

PROcapture Commands

Printed in U.S.A.

Chapter 9

PROcapture Commands

This chapter lists and briefly describes all the commands available in PROcapture. They are listed in alphabetical order. It also lists the PROcapture toolbar icons. For a more detailed description of these commands, refer to the PRO Series documentation from Viewlogic.

For a description of the fields that are common to many PRO Series dialog boxes, such as OK and Cancel, see the "Dialog Boxes" section of the "Design Entry" chapter.

Menus

The nine PROcapture menus are described in the following sections. You can select these commands from the keyboard by typing the underlined letter in each command.

File Menu

The File menu contains commands that open, close, save, and print a schematic file, and exit PROcapture. The commands on the File menu are the following:

Open	Opens a schematic or symbol sheet
Close	Closes a schematic or symbol sheet
Save	Saves the schematic or symbol to a file
Save As	Saves the schematic or symbol to another file
Print	Sends the schematic to the default printer
Exit	Exits PROcapture

Edit Menu

The Edit menu contains commands that allow you to manipulate and change the objects in your schematic. The commands on the Edit menu are the following:

Undo	Reverses effect of most recently executed command
Cut	Removes selected objects in the active window to the clipboard
Copy	Copies selected objects in the active window to the clipboard
Paste	Copies contents of the clipboard to the active window
Clear	Removes selected objects from the active window
Move	Moves selected objects to a new location
Select	Identifies objects to be operated on by the next command
Find	Searches the design hierarchy for nets, pins, or components

View Menu

The View menu contains commands that control a schematic's visibility. The commands on the View menu are the following:

In	Increases magnification of a symbol or schematic
Out	Decreases magnification of a symbol or schematic
Full	Displays the entire view of a symbol or schematic
Refresh	Redraws the active window
Region	Increases magnification in the specified area
Selected	Increases magnification of selected objects
Push Into Schematic	Pushes into a schematic of a symbol
Push Into Symbol	Pushes into a symbol
Push Next Sheet	Displays the next sheet of the schematic in sequential order
Pop	Returns to the view before execution of the last Push command

PROwave Annotation	Displays back-annotation values from the PROwave annotation file
Toolbar	Sets the toolbar display

Add Menu

The Add menu contains commands that allow you to add objects to your schematic. The commands on the Add menu are the following:

Component	Adds a component to the active schematic
Net	Adds a net to the active schematic
Bus	Adds a bus to the active schematic
Pin	Adds a pin to the active symbol
Line	Adds a line to the active schematic
Box	Adds a box to the active schematic
Circle	Adds a circle to the active schematic
Arc	Adds an arc to the active schematic
Text	Adds text to the active schematic
Object Label	Adds identifying text to an object
Object Attribute	Adds an attribute to components, pins, nets, or buses in a schematic window or to symbol pins in a symbol window
Array	Adds an array to the active schematic or symbol

Change Menu

The Change menu contains commands that allow you to change the elements in your schematic. The commands on the Change menu are the following:

Component	Changes the selected component to another component
Pin Sense	Draws invert bubble on a pin (not valid for Xilinx designs)
Slot	Changes the position of a component in the package
Transform	Changes the angle, scale, size, shape, and reflection of a graphical object
Fill Style	Changes the fill pattern of the selected object
Text	Edits schematic text
Object Color	Changes color of the selected object

Object Attributes	Controls the addition, removal, editing, and visibility of the selected object
Object Label	Controls label properties
Sheet Size	Changes the active schematic or symbol sheet size
Symbol Type	Changes the type of a symbol in the active schematic
PROcapture Colors	Sets default colors for PROcapture window
PROcapture Parameters	Changes PROcapture properties

Tools Menu

The Tools menu contains commands that allow information to be passed between PROcapture and other programs. The commands on the Tools menu are the following:

Check	Performs an electrical and connectivity check on the design
Refdes	Invokes the REFDES program
BOM	Invokes the BOM program
LDRC	Invokes the LDRC program
XREF	Invokes the XREF program
Link to PROsim	Creates a simulation netlist file for use in PROsim
Link to Spice	Creates a circuit file for use in PROspice
To EDIF	Creates an EDIF netlist file from the active schematic
From EDIF	Reads an EDIF netlist file and generates a schematic
PCB Interfaces	Transfers design and device information between the PROcapture and PC CAD layout systems

Info Menu

The Info menu contains commands that give information about PROcapture's files, directories, settings, and elements in the open schematics. The commands on the Info menu are the following:

Status	Displays the PROcapture default settings
Object Count	Displays the number of objects of each type in the design

Object Detail	Gives detailed information about the selected object
Object Attributes	Lists the attributes attached to the selected object
Object Labels	Lists the labels attached to the selected object
Memory Directory	Lists all the schematics and symbols used in the current schematic
Symbol Directory	Lists all symbol files in the PROcapture directories
Schematic Directory	Lists all schematic files in the PROcapture directories

Window Menu

The Window menu contains commands that control the display of the schematic windows onscreen. The commands on the Window menu are the following:

Cascade	Arranges all open windows diagonally down the screen
Tile	Arranges all open windows in rows across the screen
Arrange Icons	Arranges all icons along bottom of PROcapture window
Close All	Closes all open windows in PROcapture

Help Menu

The Help menu contains commands that allow you to obtain explanations of the PROcapture commands. It also gives you information on the current PRO Series release. The commands on the Help menu are the following:

Help	Displays information about the selected command
PROhelp	Displays online product information for the current release
Hostid	Displays the identification number of your security key
About	Displays the current PRO Series version, memory capacity, printer status, and project directory

Commands

The following sections describe the commands available in the PROcapture menus.

About (Help Menu)

The About command displays the current PRO Series version, the available memory, the printer status and settings, and the current project directory.

The equivalent keyboard command is **`version`**, which can be abbreviated as **`ve`**.

Arc (Add Menu)

The Arc command places an arc in the active schematic or symbol window.

The equivalent keyboard command is **`arc`**, which can be abbreviated as **`ar`**.

Arrange Icons (Window Menu)

The Arrange Icons command arranges the icons in the PROcapture window in a row along the bottom of the main window.

Array (Add Menu)

The Array command adds an array in the active schematic or symbol window.

The equivalent keyboard command is **`array`**, which can be abbreviated as **`arr`**.

BOM (Tools Menu)

The BOM command invokes the Bill of Materials program, which creates a *project*.bom file that lists the total number of devices used in the schematic by type, the total number of all devices, reference designators, corresponding values, and device documentation numbers. *Project* is the name of the schematic.

The equivalent keyboard command is **`sys bill`**.

Box (Add Menu)

The Box command adds a box to the active schematic or symbol window.

The equivalent keyboard command is **`box`**.

Bus (Add Menu)

The Bus command adds a bus to the active schematic window.

The equivalent keyboard command is **`bus`**, which can be abbreviated as **`bu`**.

Cascade (Window Menu)

The Cascade command arranges the open windows diagonally down the screen so that they overlap one on top of another, as shown in Figure 9-1. The active window is on top.

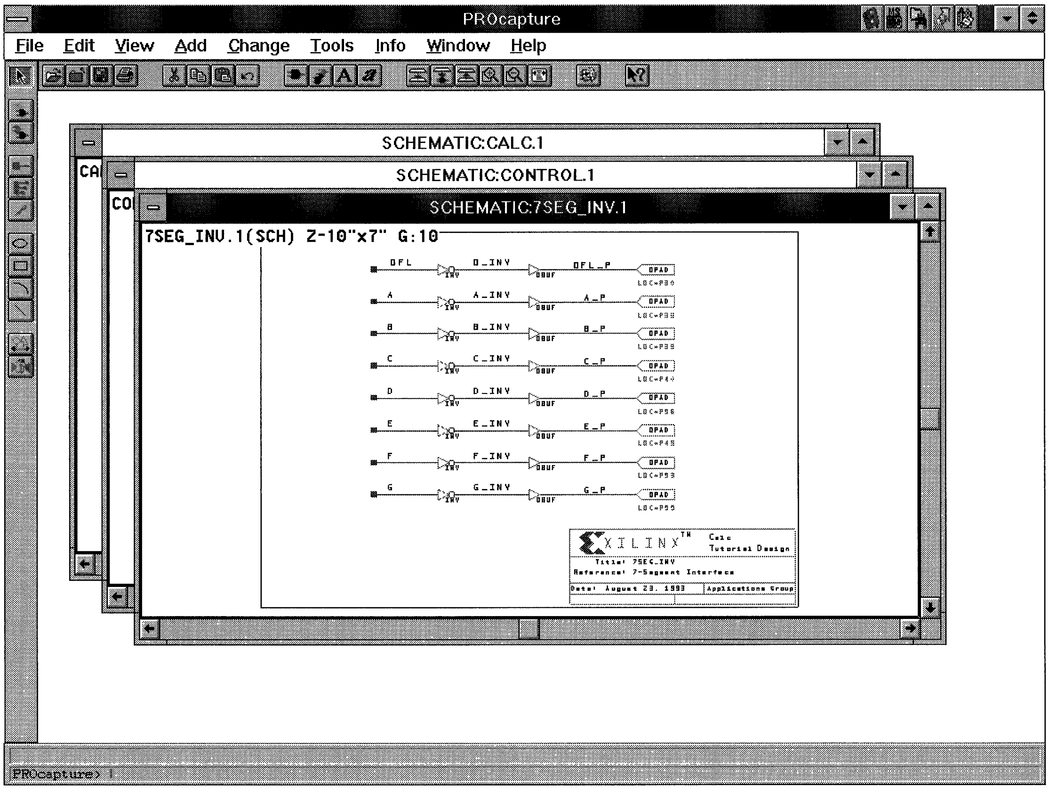

PROcapture
File Edit View Add Change Tools Info Window Help
SCHEMATIC:CALC.1
SCHEMATIC:CONTROL.1
SCHEMATIC:7SEG_INV.1
7SEG_INV.1(SCH) Z-10"x7" G:10
PROcapture>

Figure 9-1 Cascaded Windows

Check (Tools Menu)

The Check command performs electrical and connectivity checks on your design. It creates a single-sheet connectivity file for the active schematic window. The Check command has two options:

- Project checks all levels of hierarchy in the design.
- Sheet checks only the current level of hierarchy.

The equivalent keyboard command is **`check`**, which can be abbreviated as **`ch`**.

Circle (Add Menu)

The Circle command adds a circle to the active schematic or symbol window.

The equivalent keyboard command is **`circle`**, which can be abbreviated as **`ci`**.

Clear (Edit Menu)

The Clear command deletes the selected objects from the active window.

The equivalent keyboard command is **`delete`**, which can be abbreviated as **`del`**.

Close (File Menu)

The Close command closes the active schematic or symbol window.

The equivalent keyboard command is **`wclose`**, which can be abbreviated as **`wcl`**.

Close All (Window Menu)

The Close All command closes all the open windows in PROcapture.

Component (Add Menu)

The Component command on the Add menu adds a component to the active schematic window. A component is an instantiation or reference to a symbol.

The equivalent keyboard command is **`component`**, which can be abbreviated as **`com`**.

Component (Change Menu)

The Component command on the Change menu changes the selected components to another component. Except for component pin attributes, the instance attributes of the replaced component are assumed by the new component. The new component is reconnected to the same elements in the schematic if its pins are in the same position as those on the replaced component. If the pins do not match or are misaligned, PROcapture indicates that the net is a dangling net.

The equivalent keyboard command is **ccomp**, which can be abbreviated as **cc**.

Copy (Edit Menu)

The Copy command copies the selected objects into the clipboard.

The equivalent keyboard command is **bcopy**, which can be abbreviated as **bcop**.

Cut (Edit Menu)

The Cut command removes the selected objects from the window and places them in the clipboard.

The equivalent keyboard command is **bcut**, which can be abbreviated as **bcu**.

Exit (File Menu)

The Exit command exits PROcapture.

The equivalent keyboard command is **quit**, which can be abbreviated as **q**.

Fill Style (Change Menu)

The Fill Style command changes the fill pattern of the selected objects. It brings up a graphical bar in the PROcapture window from which you can select a fill style.

The equivalent keyboard command is **fstyle**, which can be abbreviated as **fs**.

Find (Edit Menu)

The Find command activates the Find dialog box, shown in Figure 9-2. It allows you to find nets, pins, and components by name, labels, or attributes. It searches through the design's hierarchy, finds the item, opens the associated schematic or symbol, centers it onscreen, and zooms it.

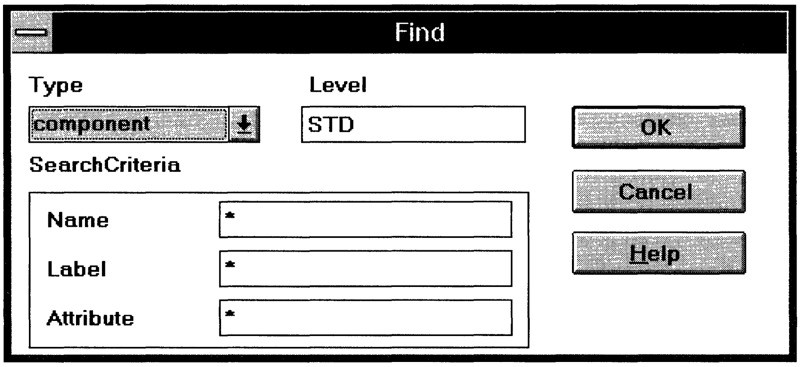


Figure 9-2 Find Dialog Box

The fields on this dialog box are the following.

- Type indicates the category of schematic entity to search for. You can select components, nets, pins, or all components. Component is the default.
- Level is the level of hierarchy to which the tool should search. STD is the default.
- Name specifies the name of the entity to search for.
- Label specifies the entity label to search for.
- Attribute specifies the entity attribute to search for.

From EDIF (Tools Menu)

The From EDIF command reads an EDIF netlist file and generates a schematic.

The equivalent keyboard command is **`mexecute edifin.mac`**, which can be abbreviated as **`mex edifin.mac`**.

Full (View Menu)

The Full command displays the entire schematic or symbol in the active window.

The equivalent keyboard command is **`full`**, which can be abbreviated as **`fu`**. You can also press the F4 key.

Help (Help Menu)

The Help command displays information about the selected command. See the "Design Entry" chapter for instructions on using this command.

Hostid (Help Menu)

The Hostid command displays the identification number of the security key found on the parallel port. This identification number also appears in the license file found in proser\standard\license.wv.

In (View Menu)

The In command increases the magnification of the schematic or symbol in the active window.

The equivalent keyboard command is **`in`**, which can be abbreviated as **`i`**. You can also press the F7 key.

LDRC (Tools Menu)

The LDRC command invokes the Logical Design Rule Check program on a design. This program checks the design for violations, fanout and fanin errors, nets without inputs or outputs, nets with multiple outputs, and partially connected components.

The equivalent keyboard command is **`system ldrc`**, which can be abbreviated as **`sys ldrc`**.

Line (Add Menu)

The Line command adds a line between two points or a series of line segments in the active schematic or symbol window.

Note: A line or line segment does not represent connectivity.

The equivalent keyboard command is **`line`**.

Link to PROsim (Tools Menu)

The Link to PROsim command creates a simulation network (VSM file) of straight schematic (non-X-BLOX) designs for use in PROsim.

The equivalent keyboard command is **`system vsm`**, which can be abbreviated as **`sys vsm`**.

Link to Spice (Tools Menu)

The Link to Spice command creates a circuit file of the schematic, which you can use as input to the PROspice analog circuit simulator. It runs the SPICELink program, wspice.exe.

The equivalent keyboard command is **`mexecute spice.mac`**, which can be abbreviated as **`mex spice.mac`**.

Memory Directory (Info Menu)

The Memory Directory command lists all the schematics and symbols used in the design. It identifies from which library they were accessed.

The equivalent keyboard command is **`dir`**.

Move (Edit Menu)

The Move command moves the selected objects to the indicated location in the active window.

The equivalent keyboard command is **`move`**, which can be abbreviated as **`mo`**.

Net (Add Menu)

The Net command adds a net that connects two component pins and/or nets in the active schematic window. It can also connect or continue a dangling net.

The equivalent keyboard command is **`net`**, which can be abbreviated as **`ne`**.

Object Attribute (Add Menu)

The Object Attribute command on the Add menu adds attribute text to components, component pins, nets, or buses in the active schematic window or to pins in a symbol window. It also adds attribute text to an unattached attribute in a schematic or symbol.

The equivalent keyboard command is **`attribute`**, which can be abbreviated as **`at`**.

Object Attributes (Change Menu)

The Object Attributes command on the Change menu controls the addition, removal, editing, and visibility of attributes attached to a selected object. It contains the following options.

- Dialog allows you to add, edit, and remove attributes associated with the selected object. If no object is selected, it lists schematic or symbol attributes that you can add, remove, and edit. Figure 9-3 gives an example of the Edit Attributes dialog box, which is displayed by this command.

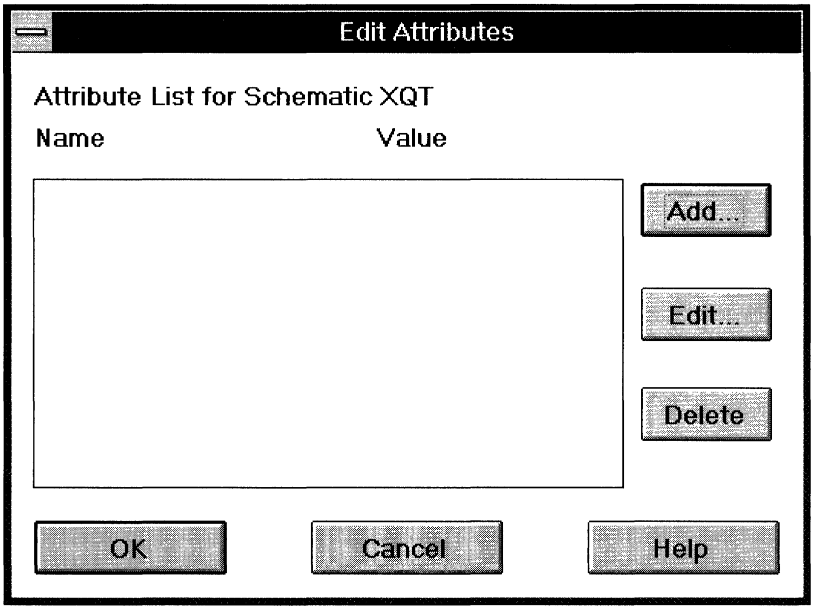


Figure 9-3 Edit Attributes Dialog Box

The Edit Attributes dialog box contains the following controls.

- Add places an attribute on a selected object. It brings up the Edit Attribute dialog box, shown in Figure 9-4.
- Edit changes the text of an attribute on a selected object.
- Delete removes an attribute from a selected object.

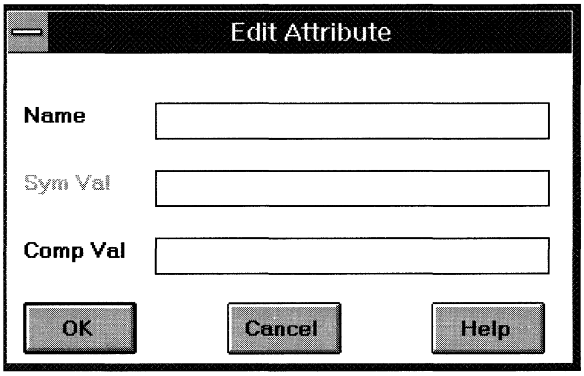


Figure 9-4 Edit Attribute Dialog Box

On the Edit Attribute dialog box, Name is the name of the attribute, Sym Val is the attribute value for a symbol definition, and Comp Val is the attribute value for an instantiation of the symbol.

The equivalent keyboard command is **`cattr`** `[`*filter*`]`, which can be abbreviated as **`ca`** `[`*filter*`]`. *Filter* is the name of a filter file that causes PROcapture to display only the attributes that you specify.

- Visibility controls the visibility of the attributes on the schematic or symbol in the active window. You can select the following options.
    - All Attrs On makes all attributes on the schematic or symbol visible in the active window.

      The equivalent keyboard command is **`avis`**, which can be abbreviated as **`avi`**.
    - All Attrs Off makes all attributes on the schematic or symbol invisible in the active window.

      The equivalent keyboard command is **`ainvis`**, which can be abbreviated as **`ai`**.
    - Attrs Name Visible makes only the name portion, not the value portion, of the specified attributes visible.

The equivalent keyboard command is **anvis**, which can be abbreviated as **anv**.

- Attrs Value Visible makes only the value portion, not the name portion, of the specified attributes visible.

  The equivalent keyboard command is **avvis**, which can be abbreviated as **avv**.

Object Attributes (Info Menu)

The Object Attributes command on the Info menu lists the attributes attached to the selected object. If an object is not selected, it lists the attributes for the current block. The "Obtaining Information" section of the "Design Entry" chapter gives an example of the information displayed.

The equivalent keyboard command is **alist**, which can be abbreviated as **al**.

Object Color (Change Menu)

The Object Color command changes the color of the selected object. It brings up a color bar in the PROcapture window from which to select a color. The color bar can display 2, 4, or 16 colors, depending on the graphics device.

The equivalent keyboard command is **color**, which can be abbreviated as **col**.

Object Count (Info Menu)

The Object Count command displays the number of objects of each type and the number of segments, joints, and connections in the active schematic or symbol window. The "Obtaining Information" section of the "Design Entry" chapter gives an example of the information displayed.

The equivalent keyboard command is **ocount**, which can be abbreviated as **oc**.

Object Detail (Info Menu)

The Object Detail command gives detailed information about the selected objects, such as orientation, color, and the component library alias. If an object is not selected, it displays information about all objects. The "Obtaining Information" section of the "Design Entry" chapter gives an example of the information displayed.

The equivalent keyboard command is **`odetail`**, which can be abbreviated as **`od`**.

Object Label (Add Menu)

The Object Label command on the Add menu adds identifying text to an object to distinguish it from similar objects. Labels differ from the text applied with the Add Text command by becoming part of the object and appearing in the netlist. Text is only annotational. You can attach a label to net segments, bus segments, symbols, pins, and components.

The equivalent keyboard command is **`label`**, which can be abbreviated as **`la`**.

Object Label (Change Menu)

The Object Label command on the Change menu governs the label properties. It includes the following options.

- Set Scope determines whether net and bus labels in the schematic are applied globally or locally.
  - Global applies the net and bus labels globally throughout the hierarchy of the schematic in the active window.

    The equivalent keyboard command is **`global`**, which can be abbreviated as **`gl`**.
  - Local applies net and bus labels locally — that is, to one level of the hierarchy — to the schematic in the active window.

    The equivalent keyboard command is **`local`**, which can be abbreviated as **`loc`**.
- Set Sense draws an invert bar over a label in the active window to indicate a logical inversion. You can only specify inversion for the entire label; you cannot specify inversion for only part of the label.

The equivalent keyboard command is **`lsense`**, which can be abbreviated as **`ls`**.

Note: Set Sense cannot be used in a Xilinx design.

- All Labels On makes the specified labels visible.

  The equivalent keyboard command is **`lvis`**, which can be abbreviated as **`lv`**.

- All Labels Off makes the specified labels invisible.

  The equivalent keyboard command is **`linvis`**, which can be abbreviated as **`linv`**.

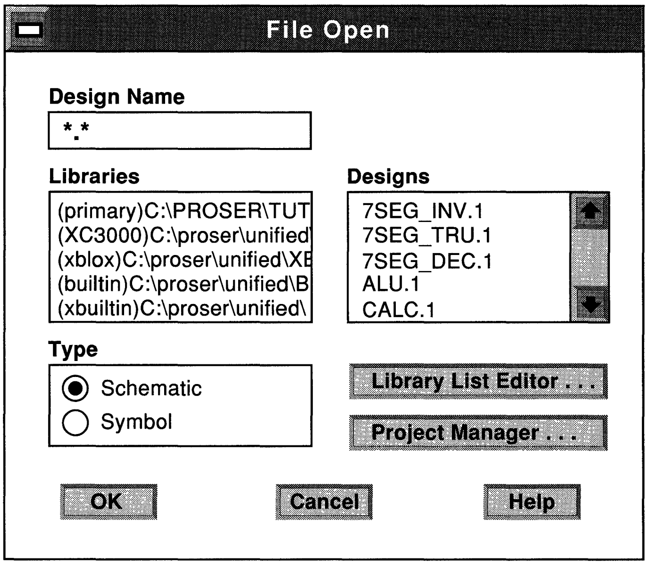
Object Labels (Info Menu)

The Object Labels command on the Info menu lists the labels attached to the selected object. If no object is selected, it lists the labels for the current block. The "Obtaining Information" section of the "Design Entry" chapter gives an example of the information displayed.

The equivalent keyboard command is **`llist`**, which can be abbreviated as **`ll`**.

Open (File Menu)

The Open command on the File menu opens a schematic or symbol sheet in the PROcapture window. It activates the File Open dialog box, shown in Figure 9-5.



Design Name
.

Libraries
(primary)C:\PROSER\TUT
(XC3000)C:\proser\unified
(xblox)C:\proser\unified\XE
(builtin)C:\proser\unified\B
(xbuiltin)C:\proser\unified\

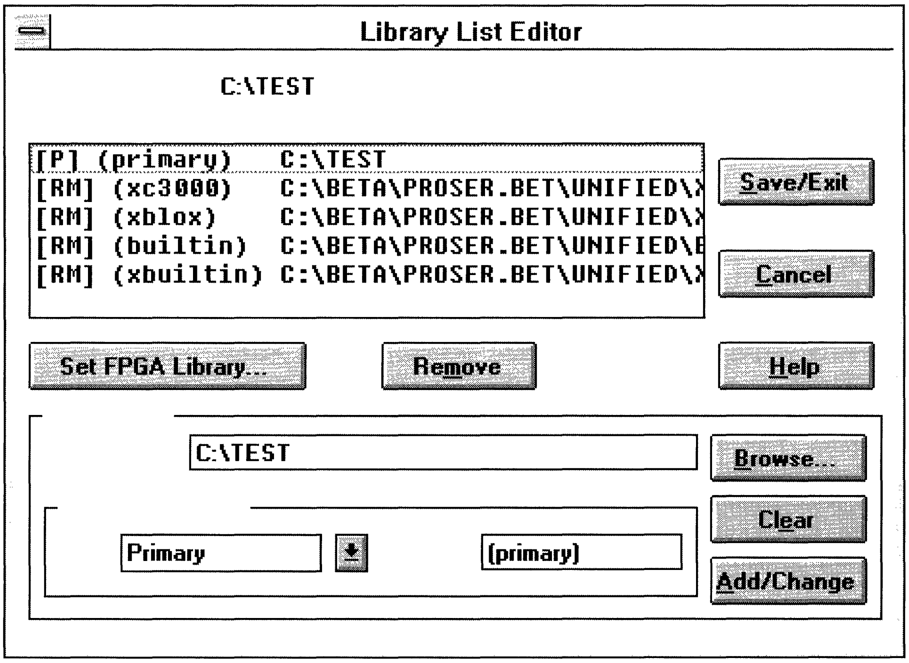
Designs
7SEG_INV.1
7SEG_TRU.1
7SEG_DEC.1
ALU.1
CALC.1

Type
[✓ Schematic]
Symbol

Figure 9-5 File Open Dialog Box

The dialog box contains the following fields.

- Design Name specifies the name of the design to open.
- Libraries lists the search paths of the libraries from which to select components.
- Designs lists the schematics available in the library highlighted in the Libraries field.
- Type indicates whether the sheet is a schematic or symbol sheet.
- Library List Editor activates the Library List Editor, shown in Figure 9-7, which enables you to add, change, and remove libraries in your directory search path. If a schematic is already open in the PROcapture window, this option is grayed out. The Library List Editor options are described in the "Getting Started" chapter.



```
[P] (primary)   C:\TEST
[RM] (xc3000)   C:\BETA\PROSER.BET\UNIFIED\
[RM] (xblox)    C:\BETA\PROSER.BET\UNIFIED\
[RM] (builtin)  C:\BETA\PROSER.BET\UNIFIED\
[RM] (xbuiltin) C:\BETA\PROSER.BET\UNIFIED\
```

C:\TEST

Primary

(primary)

Figure 9-6 Library List Editor

- Project Manager activates the PRO Series Project Manager, shown in Figure 9-7, which allows you to create a project. If a schematic is already open in the PROcapture window, this option is grayed out.

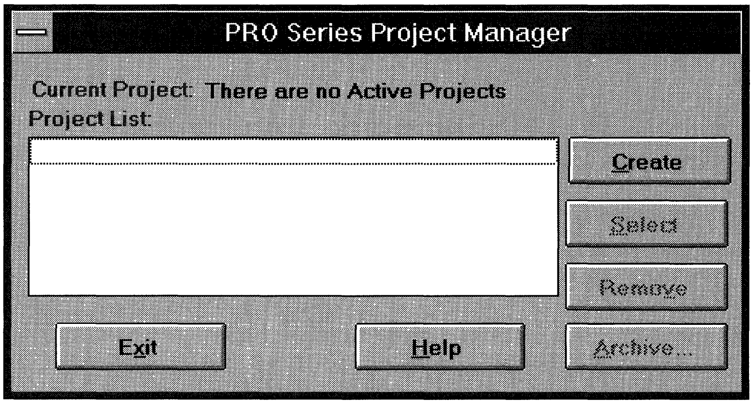

PRO Series Project Manager
Current Project: There are no Active Projects
Project List:
Create
Select
Remove
Exit
Help
Archive...

Figure 9-7 PRO Series Project Manager

The equivalent keyboard command is **schematic** *design_name*, which can be abbreviated as **sch** *design_name*.

Out (View Menu)

The Out command decreases the magnification of the schematic or symbol in the active window.

The equivalent keyboard command is **out**, which can be abbreviated as **ou**. You can also press the F8 key.

Paste (Edit Menu)

The Paste command copies the contents of the clipboard to the indicated location in the active window.

The equivalent keyboard command is **bpaste**, which can be abbreviated as **bpas**.

PCB Interfaces (Tools Menu)

The PCB Interfaces command transfers design and device information between PROcapture and the specified PCB CAD layout system. It also back-annotates pin swaps, gate swaps, and reference designator changes. You can interface to and from the following systems.

- Allegro
- Boardstation
- Cadnetix
- Cadstar
- PADS
- PCAD
- Scicards
- Visula

See the PRO Series documentation from Viewlogic for the equivalent menu commands.

Pin (Add Menu)

The Pin command adds a pin to a symbol in the active symbol window.

The equivalent keyboard command is **`pin`**.

Pin Sense (Change Menu)

The Pin Sense command draws an invert bubble on the inside of the pin in the active symbol window to indicate a logical inversion. The invert bubble is purely graphical and has no logical meaning.

The equivalent keyboard command is **`psense`**, which can be abbreviated as **`psen`**.

Note: Do not use Pin Sense in Xilinx designs.

Pop (View Menu)

The Pop command returns the view in the active window to the level of hierarchy it displayed before the last Push Into Schematic, Push Into Symbol, or Push Next Sheet command was issued.

The equivalent keyboard command is **`pop`**, which can be abbreviated as **`po`**.

Print (File Menu)

The Print command sends the schematic or symbol in the active window to the default printer.

The equivalent keyboard command is **`q2_file_print`**.

PROcapture Colors (Change Menu)

The PROcapture Colors command sets the colors of the objects in the PROcapture window. It activates the PROcolor Manager dialog box, shown in Figure 9-8.

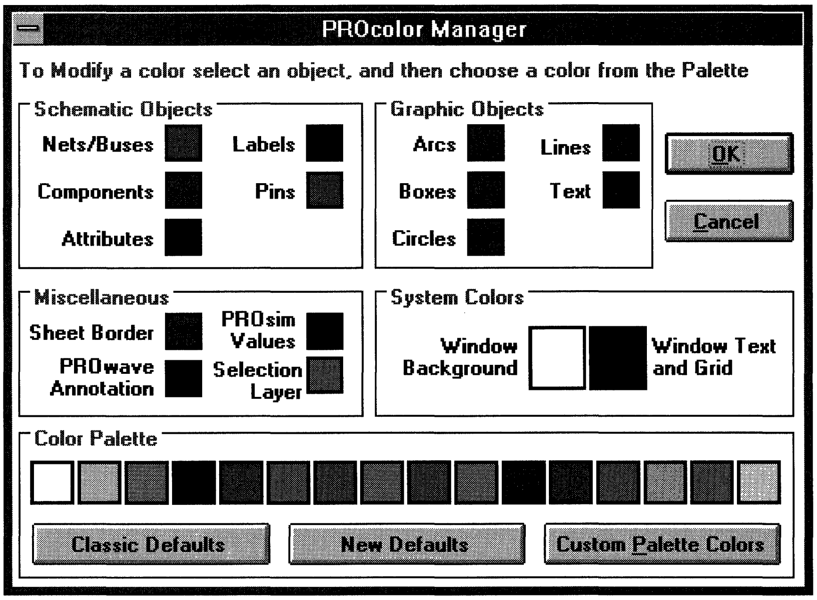


Figure 9-8 PROcolor Manager Dialog Box

This dialog box includes the following fields.

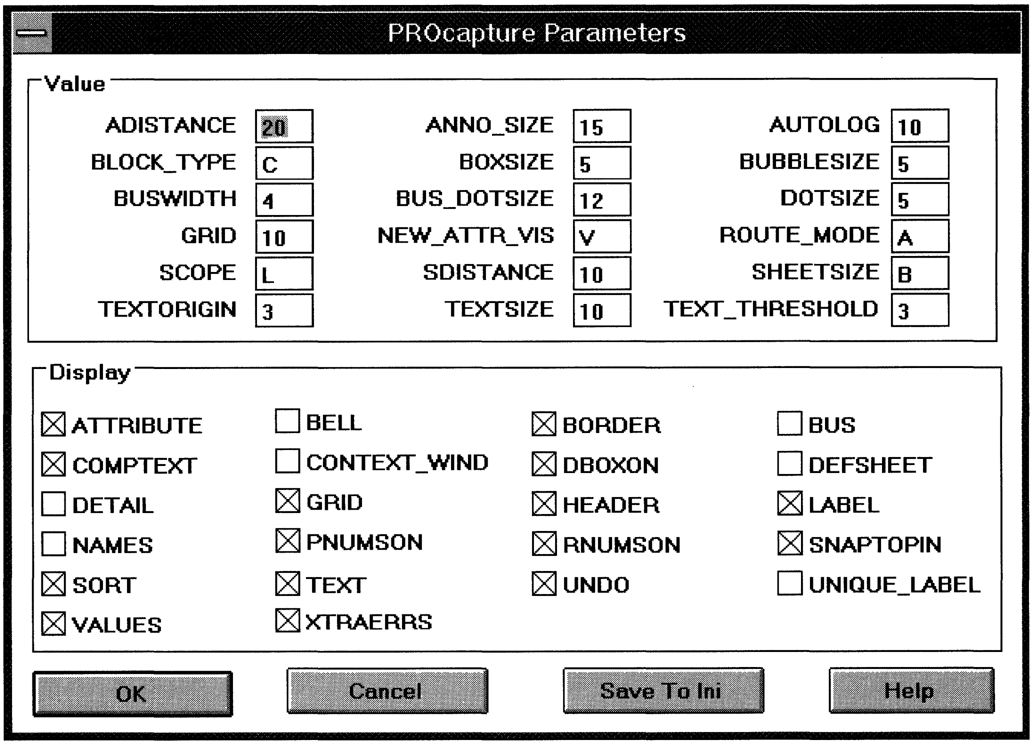
- Schematic Objects sets the color assigned to each type of schematic entity.
- Graphic Objects sets the color assigned to each type of graphical entity.

- Miscellaneous sets the colors of the sheet border, back-annotation and simulation values for PROwave and PROsim, and the selection layer. The selection layer refers to the color of the objects selected.
- System Colors determines the color of the PROcapture schematic and symbol window's background, text, and grid dots.
- Classic Defaults uses the Workview set of default colors, which includes a black background and white components.
- New Defaults uses the PRO Series set of default colors, which includes a white background and black components.
- Custom Palette Colors allows you to select the colors that appear in the Color Palette field.

The equivalent keyboard command is **`setcolor`**.

PROcapture Parameters (Change Menu)

The PROcapture Parameters command allows you to change the display properties of the PROcapture window. It activates the PROcapture Parameters dialog box, shown in Figure 9-9.


PROcapture Parameters

Value

ADISTANCE	20	ANNO_SIZE	15	AUTOLOG	10
BLOCK_TYPE	C	BOXSIZE	5	BUBBLESIZE	5
BUSWIDTH	4	BUS_DOTSIZE	12	DOTSIZE	5
GRID	10	NEW_ATTR_VIS	V	ROUTE_MODE	A
SCOPE	L	SDISTANCE	10	SHEETSIZE	B
TEXTORIGIN	3	TEXTSIZE	10	TEXT_THRESHOLD	3

Display

☒ ATTRIBUTE	☐ BELL	☒ BORDER	☐ BUS
☒ COMPTEXT	☐ CONTEXT_WIND	☒ DBOXON	☐ DEFSHEET
☐ DETAIL	☒ GRID	☒ HEADER	☒ LABEL
☐ NAMES	☒ PNUMSON	☒ RNUMSON	☒ SNAPTOPIN
☒ SORT	☒ TEXT	☒ UNDO	☐ UNIQUE_LABEL
☒ VALUES	☒ XTRAERRS		

Figure 9-9 PROcapture Parameters Dialog Box

See the PRO Series documentation from Viewlogic for an explanation of the fields of this dialog box.

The Save to Ini button saves the changes to the viewdraw.ini file in the current project.

The equivalent keyboard command is **cparams**, which can be abbreviated as **cpa**.

PROhelp (Help Menu)

The PROhelp command displays online product information for the current PROcapture release. You can obtain information on release notes, installation procedures, software problem reports, technical reference information, technical support services, and product data sheets and pricing.

PROwave Annotation (View Menu)

The PROwave Annotation command, when turned on, reads the PROwave annotation file and displays the back-annotation values in PROcapture. When turned off, the back-annotation values are not displayed in PROcapture. By default, these values are turned on.

The equivalent keyboard command is **`annoon`** to turn the annotation on and **`annooff`** to turn the annotation off. By default, annotation is turned on.

Push Into Schematic (View Menu)

The Push Into Schematic command pushes into the schematic definition of the selected symbol. Because it pushes down only one level, you may need to select this command repeatedly to descend through several levels of hierarchy.

The equivalent keyboard command is **`psch`**, which can be abbreviated as **`psc`**.

Push Into Symbol (View Menu)

The Push Into Symbol command pushes into the symbol definition of the selected symbol.

The equivalent keyboard command is **`psym`**, which can be abbreviated as **`psy`**.

Push Next Sheet (View Menu)

The Push Next Sheet command displays the sheets of a schematic or symbol in sequential order. It displays the first sheet, then the second, then the third, and so forth.

The equivalent keyboard command is **psheet**, which can be abbreviated as **psh**.

Refdes (Tools Menu)

The Refdes command invokes the REFDES program, which places components into packages by assigning reference designators to a project. Reference designators are labels that indicate to which chip a component is assigned.

The equivalent keyboard command is **mex rfdes.mac**.

Refresh (View Menu)

The Refresh command restores the view in the active window.

The equivalent keyboard command is **refresh**, which can be abbreviated as **refr**. You can also press the F5 key.

Region (View Menu)

The Region command increases the magnification in the area that you specify in the active window.

The equivalent keyboard command is **zoom**, which can be abbreviated as **zo**. You can also press the F9 key.

Save (File Menu)

The Save command saves the active schematic or symbol to a file. As it saves, PROcapture automatically performs the Tools → Check command to check the design.

The equivalent keyboard command is **write**, which can be abbreviated as **wri**.

Save As (File Menu)

The Save As command saves the schematic or symbol in the active window to another file. PROcapture prompts you to specify the block name and the sheet number. If the file already exists, PROcapture prompts you to verify that you want to overwrite it.

The equivalent keyboard command is **`writeto`**, which can be abbreviated as **`writet`**.

Schematic Directory

The Schematic Directory command lists all the schematic files in the PROcapture directories as specified in the viewdraw.ini file.

The equivalent keyboard command is **`dirsch`**, which can be abbreviated as **`dirsc`**.

Select (Edit Menu)

The Select command identifies the objects to be operated on by the next command. It offers the following options.

- Group selects a group of objects.

  The equivalent keyboard command is **`sgroup`**, which can be abbreviated as **`sga`**.

- Label selects an object label.

  The equivalent keyboard command is **`slabel`**, which can be abbreviated as **`sla`**.

- Attribute selects an attribute.

  The equivalent keyboard command is **`sattr`**, which can be abbreviated as **`sat`**.

- Text selects text.

  The equivalent keyboard command is **`stext`**, which can be abbreviated as **`ste`**.

- By Name selects a labeled net, component, or pin.

  The equivalent keyboard command is **`sname`**, which can be abbreviated as **`sna`**.

- By Value selects a net, component, or pin by the specified attribute value.

  The equivalent keyboard command is **`svalue`**, which can be abbreviated as **`sva`**.

- Component selects all instantiations of the specified symbol name.

  The equivalent keyboard command is **scomp**, which can be abbreviated as **sco**.

- Unselect deselects an object.

  The equivalent keyboard command is **unselect**, which can be abbreviated as **uns**.

Selected (View Menu)

The Selected command increases the magnification of selected objects.

The equivalent keyboard command is **zselect**, which can be abbreviated as **zse**.

Sheet Size (Change Menu)

The Sheet Size command changes the schematic or symbol sheet size in the active window. You can change it to any of the following sizes:

- A — 8 1/2 by 11 inches

  The equivalent keyboard command is **asize**, which can be abbreviated as **asi**.

- B — 17 by 11 inches

  The equivalent keyboard command is **bsize**, which can be abbreviated as **bs**.

- C — 17 by 22 inches

  The equivalent keyboard command is **csize**, which can be abbreviated as **cs**.

- D — 34 by 22 inches

  The equivalent keyboard command is **dsize**, which can be abbreviated as **ds**.

- E — 34 by 44 inches

  The equivalent keyboard command is **esize**, which can be abbreviated as **es**.

- A0 — 1189 by 841 millimeters (A0 metric sheet)

  The equivalent keyboard command is **`a0size`**, which can be abbreviated as **`ao`**.
- A1 — 594 by 841 millimeters (A1 metric sheet)

  The equivalent keyboard command is **`a1size`**, which can be abbreviated as **`a1`**.
- A2 — 594 by 420 millimeters (A2 metric sheet)

  The equivalent keyboard command is **`a2size`**, which can be abbreviated as **`a2`**.
- A3 — 420 by 297 millimeters (A3 metric sheet)

  The equivalent keyboard command is **`a3size`**, which can be abbreviated as **`a3`**.
- A4 — 210 by 297 millimeters (A4 metric sheet)

  The equivalent keyboard command is **`a4size`**, which can be abbreviated as **`a4`**.
- Z-WxH — Prompts you to specify a width and a height in units of 0.01 inch.

  The equivalent keyboard command is **`zsize`**, which can be abbreviated as **`zsi`**.

You can specify the default sheet size using the BLOCK_TYPE option in the PROcapture Parameters dialog box.

Slot (Change Menu)

The Slot command changes the position, or slot, of the selected component in the package. It also updates the pin numbers and reference designator information.

The equivalent keyboard command is **`slot`**, which can be abbreviated as **`slo`**.

Status (Info Menu)

The Status command displays the default settings for the PROcapture modes, values, routing, and directories.

The equivalent keyboard command is **`status`**, which can be abbreviated as **`sta`**.

Symbol Directory (Info Menu)

The Symbol Directory command lists all the symbol files in the PROcapture directories as specified in the viewdraw.ini file.

The equivalent keyboard command is **`dirsym`**, which can be abbreviated as **`dirsy`**.

Symbol Type (Change Menu)

The Symbol Type command changes the type of symbol placed in a schematic.

- Composite changes the symbol block in the active window to a composite block. It instructs the Tools → Check command to trace down the block hierarchy to see if there is a schematic beneath the symbol. This is the default block type.

  The equivalent keyboard command is **`bcomposite`**, which can be abbreviated as **`bcom`**.

- Module changes the symbol block in the active window to a module block, which instructs the wirelist program to stop at that level for that block in the hierarchy.

  The equivalent keyboard command is **`bmodule`**, which can be abbreviated as **`bm`**.

- Annotate changes the symbol block in the active window to an annotation block, which allows you to place documentation graphics such as title boxes in a schematic. It instructs the wirelist program or the Tools → Link to PROsim command to ignore the annotation block when it is placed as a component in a schematic.

  The equivalent keyboard command is **`bannotate`**, which can be abbreviated as **`ban`**.

- Pin changes the symbol block in the active window to a pin block, which represents a schematic pin.

  The equivalent keyboard command is **`bpin`**, which can be abbreviated as **`bpi`**.

Text (Add Menu)

The Text command on the Add menu places notational text in the active schematic or symbol window. Text is not associated with connectivity data.

The equivalent keyboard command is **`text`**, which can be abbreviated as **`tex`**.

Text (Change Menu)

The Text command on the Change menu edits the selected text in the active schematic or symbol window, including the text of attributes and labels.

The equivalent keyboard command is **`string`**, which can be abbreviated as **`stri`**.

Tile (Window Menu)

The Tile command arranges the open windows in rows across the PROcapture screen. The windows are fitted into all available screen space without overlapping. The "Copying Schematics" section of the "Design Entry" chapter gives an example of tiled windows.

To EDIF (Tools Menu)

The To EDIF command creates an EDIF netlist file of the currently open schematic.

The equivalent keyboard command is **`mexecute edifout.mac`**, which can be abbreviated as **`mex edifout.mac`**.

Toolbar (View Menu)

The Toolbar command determines which icons are displayed in the toolbar of the PROcapture window. It brings up the Toolbar dialog box, shown in Figure 9-10. See the "Toolbar Icons" section of this chapter for a description of most of the icons on the toolbar.

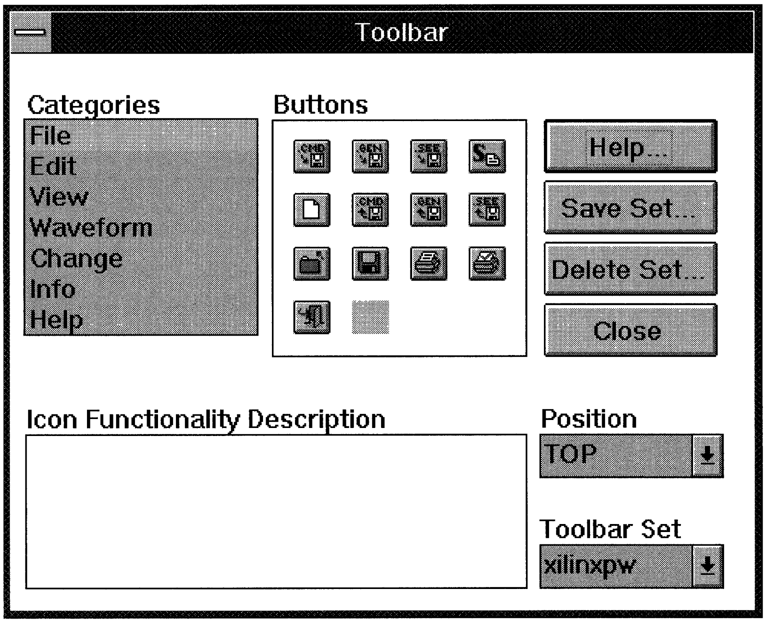


Figure 9-10 Toolbar Dialog Box

The following fields appear on this dialog box.

- Categories determines the type of icons that appear in the Buttons field. For example, if you select File, only the icons that govern files, such as the Print icon or the Open icon, appear in the Buttons box.
- Buttons displays the category of icon selected in the Categories list box. You can add an icon to the toolbar by clicking on the icon and dragging it to the toolbar. Similarly, you can delete an icon from the toolbar by dragging it into the Buttons field. When you click on an icon in this field, a brief description of its function appears in the Icon Functionality Description field.
- Save Set saves the configuration of icons in the toolbar that you arranged by dragging icons from the Buttons field.
- Delete Set removes the set of icons saved with the Save Set option.
- Close closes the dialog box without effecting any changes.

- Position governs the placement of the toolbar on the PROcapture screen. You can place it on the top, bottom, left-hand side, or right-hand side of the screen.
- Toolbar Set loads the set of icons that you saved with the Save Set command.

Transform (Change Menu)

The Transform command allows you to manipulate graphical objects. It offers the following options.

- Rotate rotates the selected object 90 degrees counterclockwise about the specified point.

  The equivalent keyboard command is **`rotate`**, which can be abbreviated as **`ro`**.
- Reflect mirrors the selected object through the specified axis.

  The equivalent keyboard command is **`reflect`**, which can be abbreviated as **`refl`**.
- Scale adjusts the size of the selected object by the value that you specify while keeping its original proportions intact.

  The equivalent keyboard command is **`scale`**, which can be abbreviated as **`sca`**.
- Stretch changes the shape and size of the selected object in any direction.

  The equivalent keyboard command is **`stretch`**, which can be abbreviated as **`stre`**.

Undo (Edit Menu)

The Undo command returns the data in the active window to the state it was in before the execution of the most recent command that modified the data.

The equivalent keyboard command is **`undo`**, which can be abbreviated as **`u`**.

XREF (Tools Menu)

The XREF command invokes the XREF program, which creates a sheet-to-sheet cross-reference report on the nets and components used in the design. It also back-annotates net attributes to the schematic.

The equivalent keyboard command is **`system xref`**, which can be abbreviated as **`sys xref`**.

Toolbar Icons

You can access some of the PROcapture commands through toolbar icons. The PROcapture window displays the following icons. When you move the cursor over each icon, a brief description of its function appears on the PROcapture command line.

Clear Icon

The Clear icon, shown in Figure 9-11, terminates the currently selected command. It is equivalent to pressing the Escape key.

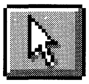


Figure 9-11 Clear Icon

Open Icon

The Open icon, shown in Figure 9-12, opens a PROcapture window so you can edit a schematic or symbol sheet. It activates the File Open dialog box, which is described in the "Commands" section of this chapter. It is equivalent to the File → Open command.

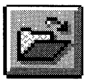


Figure 9-12 Open Icon

Close Icon

The Close icon, shown in Figure 9-13, closes the PROcapture window opened with the File → Open command. It is equivalent to the File → Close command.

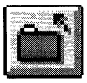


Figure 9-13 Close Icon

Save As Icon

The Save As icon, shown in Figure 9-14, saves the current schematic or symbol to another file. PROcapture prompts you to specify the block name and the sheet number. If the file already exists, PROcapture prompts you to verify that you want to overwrite it. This command is equivalent to the File → Save As command.

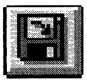


Figure 9-14 Save As Icon

Print Icon

The Print icon, shown in Figure 9-15, prints the schematic or symbol in the active window. It is equivalent to the File → Print command.

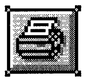


Figure 9-15 Print Icon

Cut Icon

The Cut icon, shown in Figure 9-16, removes the selected objects or areas from the window into the clipboard. It is equivalent to the Edit → Cut command.

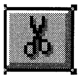


Figure 9-16 Cut Icon

Edit Copy Icon

The Edit Copy icon, shown in Figure 9-17, copies the selected objects into the clipboard. It is equivalent to the Edit → Copy command.

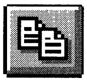


Figure 9-17 Edit Copy Icon

Paste Icon

The Paste icon, shown in Figure 9-18, copies the contents of the clipboard to the indicated location in the active window. It is equivalent to the Edit → Paste command.

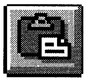


Figure 9-18 Paste Icon

Undo Icon

The Undo icon, shown in Figure 9-19, returns the data in the active window to the state it was in before the execution of the most recent command that modified the data. It is equivalent to the Edit → Undo command.

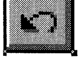


Figure 9-19 Undo Icon

Component Icon

The Component icon, shown in Figure 9-20, adds a component to the schematic in the active window. It is equivalent to the Add → Component command.

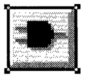


Figure 9-20 Component Icon

Array Icon

The Array icon, shown in Figure 9-21, adds an array in the active schematic window. It is equivalent to the Add → Array command.

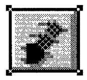


Figure 9-21 Array Icon

Text Icon

The Text icon, shown in Figure 9-22, places notational text in the active schematic or symbol window. Text is not associated with graphical or connectivity data. This icon is equivalent to the Add → Text command.

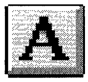


Figure 9-22 Text Icon

Attribute Icon

The Attribute icon, shown in Figure 9-23, allows you to add, edit, and remove attributes associated with a selected object. If no object is selected, it lists schematic or symbol attributes that you can add, remove, and edit. It activates the Edit Attributes dialog box, which is described in the "Object Attributes (Change Menu)" section of this chapter. This command is equivalent to the Change → Object Attributes → Dialog command.

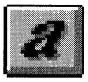


Figure 9-23 Attribute Icon

Push Into Schematic Icon

The Push Into Schematic icon, shown in Figure 9-24, pushes through a symbol into its schematic block. Because it pushes down only one level, you may need to select this command repeatedly to descend through several levels of hierarchy. It is equivalent to the View → Push Into Schematic command.

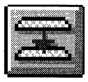


Figure 9-24 Push Into Schematic Icon

Push Into Symbol Icon

The Push Into Symbol icon, shown in Figure 9-25, pushes into a symbol block. It is equivalent to the View → Push Into Symbol command.

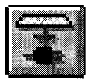


Figure 9-25 Push Into Symbol Icon

Pop Schematic Icon

The Pop Schematic icon, shown in Figure 9-26, returns the view in the active window to the level of hierarchy it displayed before the last Push Into Schematic, Push Into Symbol, or Push Next Sheet command was issued. It is equivalent to the View → Pop command.

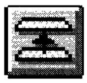


Figure 9-26 Pop Schematic Icon

In Icon

The In icon, shown in Figure 9-27, increases the magnification of the schematic or symbol in the active window. It is equivalent to the View → In command or pressing the F7 function key.

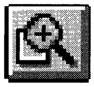


Figure 9-27 In Icon

Out Icon

The Out icon, shown in Figure 9-28, decreases the magnification of the schematic or symbol in the active window. It is equivalent to the View → Out command or pressing the F8 function key.

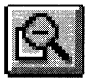


Figure 9-28 View Out Icon

Full Icon

The Full icon, shown in Figure 9-29, displays the entire schematic or symbol in the active window. It is equivalent to the View → Full command or pressing the F4 function key.

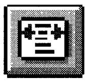


Figure 9-29 Full Icon

Help Icon

The Help icon, shown in Figure 9-30, displays information about a menu command or option when you click on it. It is equivalent to the Help → Help command. See the "Obtaining Help" section of the "Design Entry" chapter for more information.

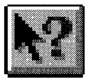


Figure 9-30 Help Icon

Move Icon

The Move icon, shown in Figure 9-31, moves the selected objects to the indicated location in the active window. It is equivalent to the Edit → Move command.

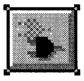


Figure 9-31 Move Icon

Copy Icon

The Copy icon, shown in Figure 9-32, copies an object to the schematic without copying to the clipboard. It is equivalent to the Copy command on the keyboard. It has no equivalent menu command.

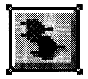


Figure 9-32 Copy Icon

Net Icon

The Net icon, shown in Figure 9-33, adds a net that connects two component pins and/or nets in the active schematic window. It can also connect or continue a dangling net. It is equivalent to the Add → Net command.

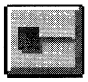


Figure 9-33 Net Icon

Bus Icon

The Bus icon, shown in Figure 9-34, adds a bus to the active schematic window. It is equivalent to the Add → Bus command.

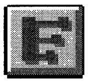


Figure 9-34 Bus Icon

Pin Icon

The Pin icon, shown in Figure 9-35, adds a pin to a symbol in the active symbol window. It is equivalent to the Add → Pin command.

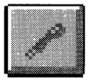


Figure 9-35 Pin Icon

Circle Icon

The Circle icon, shown in Figure 9-36, adds a circle to the active schematic or symbol window. It is equivalent to the Add → Circle command.

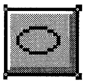


Figure 9-36 Circle Icon

Box Icon

The Box icon, shown in Figure 9-37, adds a box to the active schematic or symbol window. It is equivalent to the Add → Box command.

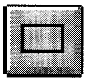


Figure 9-37 Box Icon

Arc Icon

The Arc icon, shown in Figure 9-38, places an arc in the active schematic or symbol window. It is equivalent to the Add → Arc command.

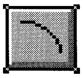


Figure 9-38 Arc Icon

Line Icon

The Line icon, shown in Figure 9-39, adds a line between two points or a series of line segments in the active schematic or symbol window. It is equivalent to the Add → Line command.

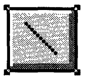


Figure 9-39 Line Icon

Stretch Icon

The Stretch icon, shown in Figure 9-40, changes the shape and size of the selected object in any direction. It is equivalent to the Change → Transform → Stretch command.

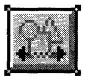


Figure 9-40 Stretch Icon

Reflect Icon

The Reflect icon, shown in Figure 9-41, mirrors the selected object through the specified axis. It is equivalent to the Change → Transform → Reflect command.

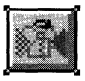


Figure 9-41 Reflect Icon

Viewlogic Interface Guide

PROsim Commands

Printed in U.S.A.

Chapter 10

PROsim Commands

This chapter lists and briefly describes all the commands available in PROsim. They are listed in alphabetical order. It also lists the PROsim toolbar icons. For a more detailed description of these commands, refer to the PRO Series documentation from Viewlogic.

For a description of the fields that are common to many PRO Series dialog boxes, such as OK and Cancel, see the "Dialog Boxes" section of the "Design Entry" chapter.

Menus

The six PROsim menus are described in the following sections. You can select these commands from the keyboard by typing the underlined letter in each command.

File Menu

The File menu contains commands that open and close a simulation network, and exit PROsim. The commands on the File menu are the following:

Open	Opens a network in PROsim
Close	Closes the loaded network
Exit	Exits PROsim

Edit Menu

The Edit menu contains commands that allow you to delete simulation entities and set the PROsim defaults. The commands on the Edit menu are the following.

Delete	Removes simulation entities from the simulation window
Defaults	Sets the PROsim defaults

View Menu

The View menu contains commands that control the visibility of the simulation entities. The commands on the View menu are the following:

PROwave	Opens a PROsim waveform file in PROwave
Path	Lists events up to the last event on specified nodes
Changes	Lists nodes that have changed during a specified time
Watchlist	Displays nodes in the circuit and their current states
Node/Vector	Displays the values of specified nodes and vectors
Report	Displays warning messages
Defaults	Displays the default settings
Toolbar	Sets the toolbar display

Simulate Menu

The Simulate menu contains commands that control how the simulation is run. The commands on the Simulate menu are the following:

Run	Simulates specified waveforms and patterns
Restart	Clears all scheduled events
Continue	Resumes simulation after an interruption
Cycle	Simulates one clock cycle
Sim	Simulates the current stepsize
Breakpoints	Sets and resets breakpoints
Command File	Runs the specified command file
Log File	Runs the PROsim log file

Values Menu

The Values menu contains commands that assign values to nodes and vectors for simulation. The commands on the Values menu are the following.

Setup	Sets simulation conditions
Assign	Assigns values to node or vector
Force	Forces nodes or vectors to specified values
Release	Releases nodes or vectors from their forced values
Display	Controls the information displayed in the PROsim window

Help Menu

The Help menu contains commands that allow you to obtain explanations of the PROsim commands. It also gives you information on the current PRO Series release. The commands on the Help menu are the following:

Help	Displays information about the selected command
PROhelp	Displays online product information for the current release
Hostid	Displays the identification number of your security key
About PROsim	Displays the current PRO Series version, memory capacity, printer status, and project directory

Commands

The following sections describe the commands available in PROsim's menus.

About PROsim (Help Menu)

The About PROsim command displays the current version of PRO Series, the available memory, the printer status and settings, and the current project directory.

The equivalent keyboard command is **`version`**, which can be abbreviated as **`ve`**.

Assign (Values Menu)

The Assign command assigns a value to a node or vector. You can specify a value in binary, octal, decimal, or hexadecimal radix. If the number of binary digits exceeds the width of the vector, the Assign command places zeros on the left.

The equivalent keyboard command is **assign** {*node_name* | *vector_name*} *value*, which can be abbreviated as **a** {*node_name* | *vector_name*} *value*.

Breakpoints (Simulate Menu)

The Breakpoints command terminates the simulation on a specified event or executes PROsim commands on circuit events. A circuit event is a node or vector transition from one logic level to another. Once PROsim encounters a breakpoint, it executes any commands associated with the breakpoint. The commands are not executed again until you reset the breakpoint.

The equivalent keyboard command is **break**, which can be abbreviated as **b**. This syntax has the following forms.

Use this syntax to set a breakpoint:

> **break** {*node_name* | *vector_name*} *value* **do** *cmd1* **...** *cmdn*

where *value* is the value at which the breakpoint is established, *node_name* or *vector_name* is the name of the node or vector at which the breakpoint is established, and *cmd1 ... cmdn* are the commands to be executed at the breakpoint.

Use this form to reset all breakpoints:

> **break**

Use the following form to reset a breakpoint every time a specific node or vector changes value:

> **break** {*node_name* | *vector_name*} *value*

Close (File Menu)

The Close command closes the current network in the PROsim window. If you have modified the data in the window, you are prompted to save it.

The equivalent keyboard command is **wclose**, which can be abbreviated as **wcl**.

Changes (View Menu)

The Changes command lists all the nodes that have changed during the specified time interval.

The equivalent keyboard command is **changes** *n1 n2* ... > *filename,* which can be abbreviated as **ch** *n1 n2* ... > *filename. N1* and *n2* are the nodes whose values have changed during the simulation interval that you specified. *Filename* is the name of the file to which you can direct the output of the command.

Command File (Simulate Menu)

The Command File command runs the specified command file. A prompt appears that allows you to type in the name of the command file. You do not have to specify a .cmd extension.

The equivalent keyboard command is **execute**, which can be abbreviated as **ex**.

Continue (Simulate Menu)

The Continue command resumes simulation of a Simulate → Sim, Simulate → Run, or Simulate → Cycle command that has been interrupted by an asynchronous interrupt or the issuance of a Simulate → Breakpoints command.

The equivalent keyboard command is **continue**.

Cycle (Simulate Menu)

The Cycle command simulates one clock cycle or a specified number of clock cycles set up with the Values → Setup → Clock command.

The equivalent keyboard command is **cycle** *n,* which can be abbreviated as **c** *n,* where *n* is the numbers of cycles to simulate.

Defaults (Edit Menu)

The Defaults command on the Edit menu sets the following defaults for the simulator.

- Command File turns on or off the display of the command file contents as the command file is read in by the simulator.

The equivalent keyboard command is **defaults** [–] **cmdfile**, which can be abbreviated as **de** [–] **cmdfile**. The dash turns off the display of command files during execution.

- Time turns on or off the automatic display of the simulation time after every command that causes simulation time to pass.

  The equivalent keyboard command is **defaults** [–] **time**, which can be abbreviated as **de** [–] **time**. The dash turns off the display of simulation time.

- Watch turns on or off the automatic display of the watch list after every Simulate → Sim command or every cycle of the Simulate → Run and Simulate → Cycle commands.

  The equivalent keyboard command is **defaults** [–] **watch**, which can be abbreviated as **de** [–] **watch**. The dash turns off the display of the watch list.

Defaults (View Menu)

The Defaults command on the View menu displays the current settings of the defaults set with the Edit → Defaults command.

The equivalent keyboard command is **defaults**.

Delete (Edit Menu)

The Delete command removes simulation entities from the PROsim window.

- Breakpoint removes all breakpoints from nodes or vectors.
  - Node removes the breakpoints from specified nodes or vectors.
  - All removes the breakpoints from all nodes and vectors.

  The equivalent keyboard command is the following:

  break [–] {**all** | *node_name* | *vector_name*}

- Clock removes clock waveforms from nodes or vectors.
  - Node removes clock waveforms from the specified nodes and vectors.
  - All removes clock waveforms from all nodes and vectors.

The equivalent keyboard command is the following:

`clock [-]` {**`all`** | *node_name* | *vector_name*}

- Pattern removes input patterns assigned to nodes or vectors with the Pattern command or the Values → Setup → Pattern command.
  - Node removes input patterns from specified nodes or vectors.
  - All removes input patterns from all nodes and vectors.

  The equivalent keyboard command is the following:

  `pattern [-]` {**`all`** | *node_name* | *vector_name*}
- Trace removes nodes or vectors from the trace list generated by the Values → Setup → Trace command.
  - Node removes specified nodes or vectors.
  - All removes all nodes and vectors.

  The equivalent keyboard command is the following:

  `trace [-]` {**`all`** | *node_name* | *vector_name*}
- Watch removes nodes or vectors from the internal watch list generated by the Values → Setup → Watch command.
  - Node removes specified nodes or vectors.
  - All removes all nodes or vectors.

  The equivalent keyboard command is the following:

  `watch [-]` {**`all`** | *node_name* | *vector_name*}
- Wfm removes from nodes all waveforms created with the Values → Setup → Wfm command.
  - Node removes the waveforms from specified nodes.
  - All removes the waveforms from all nodes.

  The equivalent keyboard command is the following:

  `wfm [-]` {**`all`** | *node_name* | *vector_name*}

Display (Values Menu)

The Display command controls the information displayed in the PROsim window.

- Inputs displays a list of nodes or vectors that are currently being forced to the H, L, or X logic values.

  The equivalent keyboard command is **`inputs`**.

- Stats lists circuit statistics in the PROsim window.

  The equivalent keyboard command is **`statistics`**, which can be abbreviated as **`stats`**.

- Info displays information about a specified node or set of nodes. This information includes the node's value, capacitance, a list of all receivers and drivers and their values, and any scheduled events on the node. You can use wildcards with this command.

  The equivalent keyboard command is **`info`** *node_name*.

- Stepsize displays the duration of the current simulation step or clock phase.

  The equivalent keyboard command is **`stepsize`**, which can be abbreviated as **`step`**.

- Time displays the current simulation time in the PROsim window.

  The equivalent keyboard command is **`echo simulation time`** `=` *time*, where *time* is the length of time that the simulation has been running.

- Status displays values on nodes and vectors.

  - Node displays values on specified nodes and vectors.
  - All displays values on all nodes and vectors.

  The equivalent keyboard command is **`defaults`** or **`report`**, which can be abbreviated as **`de`** or **`report`**.

Exit

The Exit command exits PROsim.

The equivalent keyboard command is **`quit`**, which can be abbreviated as **`q`**.

Force (Values Menu)

The Force command forces a node or vector to one of the following values.

Note: This command schedules values. Simulation must occur before the values are actually placed on the node.

You can release these values on internal nodes with the Values → Release command. See the listing for this command in this chapter for more information.

- L forces a node or vector to a logic 0. The node or vector can be either a circuit input or an internal node. You can use wildcards.

  The equivalent keyboard command is **l** {*node_name* | *vector_name*}.

- H forces a node or vector to a logic 1. The node or vector can be either a circuit input or an internal node. You can use wildcards.

  The equivalent keyboard command is **h** {*node_name* | *vector_name*}.

- X forces a node or vector to a logic X. The node or vector can be either a circuit input or an internal node. You can use wildcards.

  The equivalent keyboard command is **x** {*node_name* | *vector_name*}.

Help (Help Menu)

The Help command displays information about the selected command. It performs the same function as the Help command in PROcapture. See the "Design Entry" chapter for instructions on using this command.

Hostid (Help Menu)

The Hostid command displays the identification number of the security key found on the parallel port. This identification number also appears in the license file found in \proser\standard\license.wv.

Log File (Simulate Menu)

The Log File command runs a PROsim log file. The file name must have a .log extension; if it is not included, PROsim appends it. The default log file is called viewsim.log.

The equivalent keyboard command is **`execute`** *logfile.ext*.

Node/Vector (View Menu)

The Node/Vector command displays values on specified nodes and vectors. You can type in the names of the nodes and vectors at the prompt. Typing * displays the values for all nodes and vectors.

The equivalent keyboard command is **`s_d`** {*node_name* | *vector_name*}.

Open (File Menu)

The Open command opens a network (VSM file) for simulation. It activates the Open dialog box; an example is shown in Figure 10-1.

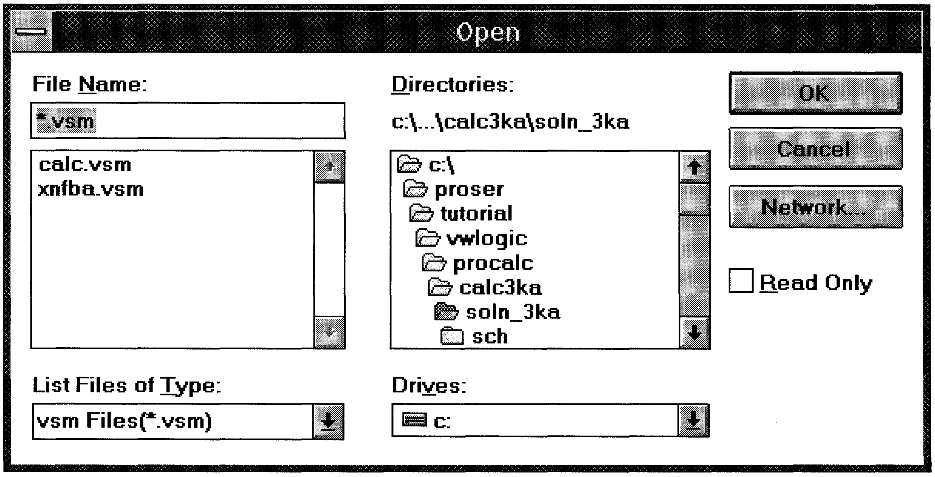


Figure 10-1 Open Dialog Box

The dialog box contains the following fields.

- File Name consists of a space for you to type in the name of the simulation network (VSM file) that you want to load (backspace over the asterisk) or a list box where you can click on the name of the design.

- List Files of Type indicates the type of file to be opened. The only available type is a VSM file.
- Directories lists the directories for the specified drive so you can select the directory that contains the VSM file that you want to load.
- Drives lists the drives to which the PC is connected so you can select the drive on which the VSM file is located.
- Network accesses the network drive so you can connect to a printer on the network. It activates the Printers - Network Connections dialog box, shown under the Printer Setup command in this chapter. Consult the Microsoft Windows documentation or click on the Help button for an explanation of the fields on this dialog box. The Network option is only available if you are running Windows for Workgroups.
- Read Only, if selected, prohibits writing to the currently loaded file.

The equivalent keyboard command is **`simu`** *network_file.*

Path (View Menu)

The Path command lists the events leading up to the last event on the specified nodes and indicates what caused these nodes to change in value. You can use this command to determine the cause of unexpected X values in the circuit.

The equivalent keyboard command is **`path`** *node_name.*

PROhelp (Help Menu)

The PROhelp command displays online product information for the current PROsim release. You can obtain information on release notes, installation procedures, software problem reports, technical reference information, technical support services, and product data sheets and pricing.

PROwave (View Menu)

The PROwave command opens or closes a PROsim waveform file in PROwave.

If you want to view waveforms in PROwave, you must issue this command before issuing any command that causes simulation time to pass, such as Run or Cycle.

The waveform file must have a .wfm extension.

The equivalent keyboard command is the following:

`wave` `[`**`stream`**`]` *filename*.**`wfm`** {*node_name1* | *vector_namen*} ...
{*node_name1* | *vector_namen*}

The Stream option outputs all simulation data to the command line.

The second form of the syntax lists all the nodes to be displayed in PROwave.

`wave` `[`**`stream`**`]` *design*.**`wfm`** < *nodes.lst*

This syntax pipes the nodes listed in *nodes.list* into the waveform file.

Release (Values Menu)

The Release command releases a node or vector from its forced value. If the node is a circuit input, it goes to Z (high impedance) unless it is subject to charge storage. In this case, it retains the forced value. Internal nodes are re-evaluated by the simulator.

The equivalent keyboard command is **`r`** *node_name*.

Report (View Menu)

The Report command displays warning messages during simulation.

The equivalent keyboard command is **`report memerrs`**. To disable the display of warning messages, type **`report -memerrs`**.

Restart (Simulate Menu)

The Restart command clears all the scheduled events, such as clocks, patterns, and breakpoints, and re-initializes the simulation time to 0. It retains all vector definitions.

The equivalent keyboard command is **`restart`**.

Run (Simulate Menu)

The Run command simulates waveforms and patterns defined by the Values → Setup → Wfm or the Values → Setup → Pattern command.

The equivalent keyboard command is **`run`** *n*, where *n* is the number of waveforms or patterns through which PROsim simulates.

Setup (Values Menu)

The Setup command establishes the conditions for simulation.

- Vector defines vectors for a node or set of nodes.

  The equivalent keyboard command is **`vector`** *vector_name node_name1 ... noden*, which can be abbreviated as **`v`** *vector_name node_name1 ... node_namen*.

- Wfm defines the stimulus or the pattern of the waveform.

  - Per generates a periodic waveform by assigning waveform values to nodes or vectors at specified times.

    The equivalent keyboard command is **`wfm`** {*node_name* | *vector_name*} **`time`**=*value* (**`time`**=*value* **`time`**=*value*)**`*`***n*.

  - Aper generates an aperiodic waveform by assigning waveform values to nodes or vectors at specified times.

    The equivalent keyboard command is **`wfm`** {*node_name* | *vector_name*} **`time`**=*value* **`time`**=*value....*

  - Inc generates a periodic sequence of patterns whose value is increased by a specified amount each period.

    The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`inc by`** *value*)**`*`***n*.

**N* specifies the number of repetitions.

Time is the starting time.

Value is the starting value.

Period is the time value between increments.

Value is the value by which to increment.

- Dec generates a periodic sequence of patterns whose value is decreased by a specified amount each period.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`dec by`** *value*)*****n.

  **N* specifies the number of repetitions.

  Time is the starting time.

  Value is the starting value.

  Period is the time value between decrements.

  Value is the value by which to decrement.

- Mult generates a periodic sequence of patterns whose value is multiplied by a specified amount each period.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`mult by`** *value*)*****n.

- Div generates a periodic sequence of patterns whose value is divided by a specified amount each period.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`div by`** *value*)*****n.

- ShftR generates a periodic sequence of patterns whose value is shifted to the right by a specified number of bits each period, the most significant bit being filled with zero.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`sr by`** *value*).

- ShftL generates a periodic sequence of patterns whose value is shifted to the left by a specified number of bits each period, the least significant bit being filled with zero.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`sl by`** *value*).

- RotR generates a periodic sequence of patterns whose value is rotated to the right by a specified number of bits each period, the most significant bit being padded with the least significant bit.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`rr by`** *value*).

- RotL generates a periodic sequence of patterns whose value is rotated to the right by a specified number of bits each period, the least significant bit being padded with the most significant bit.

  The equivalent keyboard command is **`wfm`** *vector_name time=value* (*period*=**`rl by`** *value*).

- Pattern defines a sequence of input patterns for a node or vector.

  The equivalent keyboard command is the following:

  `pattern` {*node_name* | *vector_name*} *pattern_sequence*

  Pattern_sequence is the sequence of values to put on the signal.

  It can be abbreviated as the following:

  `pat` {*node_name* | *vector_name*} *pattern_sequence*

- Clock sets the clock transitions for a specified node or vector.

  The equivalent keyboard command is the following:

  `clock` {*node_name* | *vector_name*} *value1 ... valuen*

  This option can be abbreviated as the following:

  `ck` {*node_name* | *vector_name*} *value1 ... valuen*

- Stepsize specifies the duration of the simulation step or the clock phase. The default is 100 ns.

  The equivalent keyboard command is **`stepsize`** *n*, which can be abbreviated as **`step`** *n*.

- Radix specifies the output vectors for a set of vectors. You can specify a binary, octal, decimal, or hexadecimal radix. Hexadecimal is the default.

The equivalent keyboard command is **`radix`** *base vector1 ... vectorn*, where *base* is one of the four radices just noted, and *vector1 ... vectorn* are the names of the vectors to be assigned that radix.

- Report controls the display of the warning messages issued during simulation. You can select one of the following options. The following values can all be set to on or off.
  - Spike reports the cancellation of each output spike, which PROsim normally filters out.

    The equivalent keyboard command is **`report [-] spike`**.
  - Stab displays simulation stability messages, which appear if you terminate a simulation while the values are still being propagated. These messages warn you that the results you currently see onscreen may not be the same as the final results of the simulation.

    The equivalent keyboard command is **`report [-] stab`**.
  - Time reports timing errors such as setup and hold or pulse-width violations.

    The equivalent keyboard command is **`report [-] time`**.
  - All displays all warning messages.

    The equivalent keyboard command is **`report [-] all`**.
- Print prints the values of the nodes and vectors on the watch list at specified time intervals. It prints the vector and node names in columns in the same order as they appear on the watch list.

  The equivalent keyboard command is **`print`**.
- Trace displays the old and new values of a vector or node when it changes state.

  The equivalent keyboard command is **`trace`** {*node_name* | *vector_name*}, which can be abbreviated **`t`** {*node_name* | *vector_name*}.

- Watch places nodes or vectors on the watch list, where they can be displayed and printed.

  The equivalent keyboard command is **`watch`** {*node_name* | *vector_name*}, which can be abbreviated **`w`** {*node_name* | *vector_name*}.

- Activity displays a histogram of 20 time intervals showing the circuit events within the specified time range.

  The equivalent keyboard command is **`activity`** *n1 n2*, which can be abbreviated as **`act`** *n1 n2*. *N1* is the beginning of the time range, and *n2* is the end of the time range.

- LogFile opens or closes the PROsim log file.

  The equivalent keyboard command is **`logfile`** *filename*, which can be abbreviated as **`log`** *filename*.

Sim (Simulate Menu)

The Sim command simulates the current stepsize time, which is set with the Values → Setup → Stepsize command.

The equivalent keyboard command is **`sim`** *n*, where *n* is the number of ticks. If the time units are not specified, the current stepsize is used.

Toolbar (View Menu)

The Toolbar command determines which icons are displayed in the toolbar of the PROsim window. It brings up the Toolbar dialog box, shown in Figure 10-2. See the "Toolbar Icons" section of this chapter for a description of most of the icons on the toolbar.

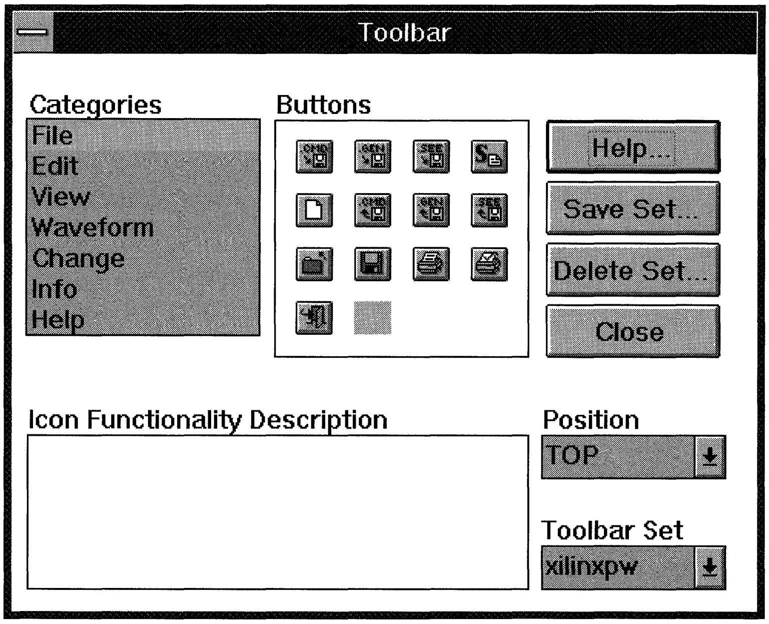


Figure 10-2 Toolbar Dialog Box

The following fields appear on this dialog box.

- Categories determines the type of icons that appear in the Buttons field. For example, if you select File, only the icons that govern files, such as the Print icon or the Open icon, appear in the Buttons box.
- Buttons displays the category of icon selected in the Categories list box. You can add an icon to the toolbar by clicking on the icon and dragging it to the toolbar. Similarly, you can delete an icon from the toolbar by dragging it into the Buttons field. When you click on an icon in this field, a brief description of its function appears in the Icon Functionality Description field.
- Save Set saves the configuration of icons in the toolbar that you arranged by dragging icons from the Buttons field.
- Delete Set removes the set of icons saved with the Save Set option.
- Close closes the dialog box without effecting any changes.

- Position governs the placement of the toolbar on the PROwave screen. You can place it on the top, bottom, left-hand side, or right-hand side of the screen.
- Toolbar Set loads the set of icons that you saved with the Save Set command.

Watchlist (View Menu)

The Watchlist command displays the nodes in the current circuit and their current states, or their logic values. Enter an asterisk (*) to display all nodes.

The equivalent keyboard command is **`display`** [*node_name* | *vector_name*], which can be abbreviated as **`d`** [*node_name* | *vector_name*].

Toolbar Icons

You can access some of the PROsim commands through toolbar icons. The PROsim window displays the following icons. When you move the cursor over each icon, a brief description of its function appears on the PROsim command line.

Open Icon

The Open icon, shown in Figure 10-3, opens a network (VSM) file for simulation. It is equivalent to the File → Open command.

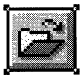


Figure 10-3 Open Icon

Exit Icon

The Exit icon, shown in Figure 10-4, exits PROsim. It is equivalent to the File → Exit command.

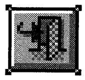


Figure 10-4 Exit Icon

Command File Icon

The Command File icon, shown in Figure 10-5, runs the specified command file. A prompt appears that allows you to type in the name of the command file. You do not have to specify a .cmd extension. This command is equivalent to the Simulate → Command File command.

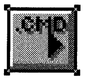


Figure 10-5 Command File Icon

Breakpoint Icon

The Breakpoint icon, shown in Figure 10-6, terminates the simulation on a specified event or executes PROsim commands on circuit events. It is equivalent to the Simulate → Breakpoints command.

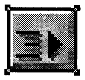


Figure 10-6 Breakpoint Icon

Continue Icon

The Continue icon, shown in Figure 10-7, resumes simulation of a Simulate → Sim, Simulate → Run, or Simulate → Cycle command that has been interrupted by an asynchronous interrupt or the issuance of a Simulate → Breakpoints command.

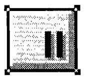


Figure 10-7 Continue Icon

Simulate Icon

The Simulate icon, shown in Figure 10-8, simulates the current stepsize time, which is set with the Values → Setup → Stepsize command. It is equivalent to the Simulate → Sim command.

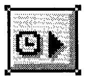


Figure 10-8 Simulate Icon

Run Icon

The Run icon, shown in Figure 10-9, simulates waveforms and patterns defined by the Values → Setup → Wfm or the Values → Setup → Pattern command. It is equivalent to the Simulate → Run command.

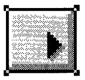


Figure 10-9 Run Icon

Cycle Icon

The Cycle icon, shown in Figure 10-10, simulates one clock cycle set up with the Values → Setup → Clock command. It is equivalent to the Simulate → Cycle command.

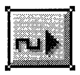


Figure 10-10 Cycle Icon

Force Low Icon

The Force Low icon, shown in Figure 10-11, forces a node or vector to a logic 0. The node or vector can be either a circuit input or an internal node. You can use wildcards. This command is equivalent to the Values → Force → L command.

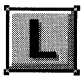


Figure 10-11 Force Low Icon

Force High Icon

The Force High icon, shown in Figure 10-12, forces a node or vector to a logic 1. The node or vector can be either a circuit input or an internal node. You can use wildcards. This command is equivalent to the Values → Force → H command.

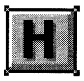


Figure 10-12 Force High Icon

Force X Icon

The Force X icon, shown in Figure 10-13, forces a node or vector to a logic X. The node or vector can be either a circuit input or an internal node. You can use wildcards. This command is equivalent to the Values → Force → X command.

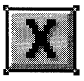


Figure 10-13 Force Logic X Icon

Help Icon

The Help icon, shown in Figure 10-14, displays information about the selected command. It is equivalent to the Help → Help command. See the "Design Entry" chapter for instructions on using this icon.

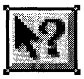


Figure 10-14 Help Icon

Viewlogic Interface Guide

PROwave Commands

Printed in U.S.A.

Chapter 11

PROwave Commands

This chapter lists and briefly describes all the commands available in PROwave. They are listed in alphabetical order. It also lists the PROwave toolbar icons. For a more detailed description of these commands and how to use them, refer to the PRO Series documentation from Viewlogic.

For a description of the fields that are common to many PRO Series dialog boxes, such as OK and Cancel, see the "Dialog Boxes" section of the "Design Entry" chapter.

Menus

The eight PROwave menus are described in the following sections. You can select these commands from the keyboard by typing the underlined letter in each command.

File Menu

The File menu contains commands that open, close, save, and print a waveform file, and exit PROwave. The commands on the File menu are the following:

New	Opens or merges a new generic output file
Open	Opens a waveform file in the PROwave window
Close	Closes a waveform file
Save	Saves the waveforms to a file
Save As	Saves the waveforms to another file
Print	Sends the waveforms to the default printer
Exit	Exits PROwave

Edit Menu

The Edit menu contains commands that allow you to manipulate and change the objects in your waveform file. The commands on the Edit menu are the following:

Cut	Removes the selected waveforms from the active window to the clipboard
Copy	Copies the specified waveform region into the clipboard
Paste	Inserts the contents of the clipboard to a point on the waveform
Insert	Inserts the specified value to the selected region of the waveform
Replace	Replaces the selected waveform region with the new value
Delete	Removes the specified waveform region
Repeat	Copies the selected waveform region and places it repeatedly
Modify Signal	Adjusts the signal display
Edit Off	Disables editing mode

View Menu

The View menu contains commands that control a waveform's visibility. The commands on the View menu are the following:

Zoom In	Increases the waveform magnification
Zoom Out	Decreases the waveform magnification
Full	Displays the entire waveform
Pan	Centers the view of the waveforms onscreen
Range	Increases the magnification in the specified area
Refresh	Restores the view in the active window
Grid	Controls the grid in the active window
Toolbar	Sets the toolbar display

Waveform Menu

The Waveform menu contains commands that control the appearance of the waveforms displayed in the PROwave window. The commands on the Waveform menu are the following.

Add	Adds a waveform to the PROwave window
Copy	Copies selected signals to new signals
Rename	Changes the name of selected signals
Remove	Deletes selected waveforms from the PROwave window
Signals	Creates, deletes, and manipulates signals
Create Text	Creates text in the PROwave window
Delete Text	Removes text from the PROwave window
Set Radix	Changes the base representation of selected bus waveforms
Display Params	Controls the waveform display

Change Menu

The Change menu contains commands that change the contents of the waveform file. The commands on the Change menu are the following:

Subst	Places the clipboard contents at the specified location on a signal
TLimit	Defines the time range in the PROwave window
TStep	Changes the time axis but not the size of the waveforms
Copy	Copies the selected waveforms into the clipboard
Cut	Removes the selected waveforms into the clipboard
Paste	Inserts the contents of the clipboard into the specified point on the waveform
Append	Adds the selected waveforms to the existing clipboard contents
Snap On	Turns on grid snapping
Snap Off	Turns off grid snapping
Annotate On	Back-annotates the simulation values to PROcapture
Annotate Off	Terminates back-annotation to PROcapture

Info Menu

The Info menu contains a command that displays waveform information. This command is the following:

Status	Displays waveform information

Window Menu

The Window menu contains commands that control the display of the waveforms onscreen. The commands on the Window menu are the following:

Cascade	Arranges all open windows diagonally down the screen
Tile	Arranges all open windows in rows across the screen
Arrange Icons	Arranges all icons along the bottom of the PROwave window
Close All	Closes all the open windows in PROwave

Help Menu

The Help menu contains commands that allow you to obtain explanations of the PROcapture commands. It also gives you information on the current PRO Series release. The commands on the Help menu are the following:

Help	Displays information about the selected command
PROhelp	Displays online release notes for the current release
Hostid	Displays the identification number of your security key
About PROwave	Displays the current PRO Series version, memory capacity, printer status, and project directory

Commands

The following sections describe the commands available in PROwave's menus.

About PROwave (Help Menu)

The About PROwave command displays the current version of PRO Series, the available memory, the printer status and settings, and the current project directory.

The equivalent keyboard command is **`version`**, which can be abbreviated as **`ve`**.

Add (Waveform Menu)

The Add command adds a waveform to the PROwave window from the status list.

The equivalent keyboard command is **`add`**, which can be abbreviated as **`ad`**.

Annotate On (Change Menu)

The Annotate On command back-annotates simulation values in PROcapture and creates a back-annotation file called *project*.ann. This command writes to the ANN file each time that you move the crosshair.

The equivalent keyboard command is **`annoon`**.

Annotate Off (Change Menu)

The Annotate Off command terminates the back-annotation to PROcapture initiated with the Change → Annotate On command.

The equivalent keyboard command is **`annooff`**.

Append (Change Menu)

The Append command adds the selected waveforms to the existing contents of the clipboard without overwriting them.

The equivalent keyboard command is **`bappend`**, which can be abbreviated as **`bap`**.

Arrange Icons (Window Menu)

The Arrange Icons command arranges the icons in the PROwave window in a row along the bottom of the window.

Cascade (Window Menu)

The Cascade command arranges the open windows diagonally down the screen so that they overlap one on top of another. The active window is on top. The "Cascade" section of the "PROcapture Commands" chapter contains an illustration of cascaded windows.

Close (File Menu)

The Close command closes the PROwave window opened with the File → Open command.

The equivalent keyboard command is **`wclose`**, which can be abbreviated as **`wcl`**.

Close All (Window Menu)

The Close All command closes all the open windows in PROwave.

Copy (Edit Menu)

The Copy command on the Edit menu copies a specified region of a waveform into the clipboard. It does not modify the signal from which the region is copied. You cannot copy multiple signals with a single Edit → Copy command.

The equivalent keyboard command is **`rcopy`**, which can be abbreviated as **`rco`**.

Copy (Waveform Menu)

The Copy command on the Waveform menu copies selected signals to new signals whose names you specify.

The equivalent keyboard command is **`copy`**, which can be abbreviated as **`co`**.

Copy (Change Menu)

The Copy command on the Change menu copies selected waveforms into the clipboard, overwriting the previous clipboard contents.

The equivalent keyboard command is **`bcopy`**, which can be abbreviated as **`bc`**.

Create Text (Waveform Menu)

The Create Text command creates text that you can place in the names region of the PROwave window. Text that you add to a waveform with this command is not linked to any signal.

The equivalent keyboard command is **`tcreate`**, which can be abbreviated as **`tcr`**.

Cut (Edit Menu)

The Cut command on the Edit menu cuts a specified region of a signal from the waveform region and places it in the clipboard. All transitions following the cut region are rigidly shifted to the left; all spaces between the transitions remain the same.

The equivalent keyboard command is **`rcut`**, which can be abbreviated as **`rcu`**.

Cut (Change Menu)

The Cut command on the Change menu cuts the selected waveforms from the names region and places them in the clipboard, overwriting the previous contents of the clipboard.

The equivalent keyboard command is **`bcut`**, which can be abbreviated as **`bc`**.

Delete (Edit Menu)

The Delete command removes a specified region of a signal. All transitions following the deleted region are rigidly shifted to the left. All spaces between the transitions remain the same. This command does not save the region in the clipboard, so it prompts you before deleting the region.

The equivalent keyboard command is **`rdelete`**, which can be abbreviated as **`rdel`**.

Delete Text (Waveform Menu)

The Delete Text command deletes the selected text strings from the names region of the PROwave window.

The equivalent keyboard command is **`tdelete`**, which can be abbreviated as **`td`**.

Display Params (Waveform Menu)

The Display Params command controls the display of the waveforms in the PROwave window. This command has the following options.

- Color changes the color of the selected waveform to the color that you specify. A color spectrum appears for you to select the new color.

  The equivalent keyboard command is **`color`**, which can be abbreviated as **`co`**.

- Notation changes the default numeric notation. You can select decimal, scientific, engineering, and units notation. This command does not change the interpretation of the waveform data in PROwave.

  The equivalent keyboard command is **`notation`**, which can be abbreviated as **`not`**.

- Values toggles the display of the values region and box in the PROwave window.

  The equivalent keyboard command is **`von`** to display the values and box, and **`voff`** to remove them from the screen.

- XHair toggles the display of the timeline and sets the crosshair to a specific location in the PROwave window.
  - The X-Hair On setting turns the timeline display on. This option is the default.

    The equivalent keyboard command is **`xhon`**.
  - X-Hair Off turns the timeline display off.

    The equivalent keyboard command is **`xhoff`**.
  - Set X-Hair allows you to specify the crosshair location.

    The equivalent keyboard command is **`setxhair`** *signal time.*
- Tunits specifies the size of the time units in a waveform and toggles the display of the time units. It does not alter the data in PROwave. These functions are set with the following options.
  - Size specifies the size of the time units.

    The equivalent keyboard command is **`tunits`**, which can be abbreviated as **`tun`**.
  - On turns the time unit display on.

    The equivalent keyboard command is **`ton`**.
  - Off turns the time unit display off.

    The equivalent keyboard command is **`toff`**, which can be abbreviated as **`tof`**.
- Widths changes the width of the values and names regions in the PROwave window. The following two options control these functions.
  - Width interactively changes the width of the values region.

    The equivalent keyboard command is **`wvalues`**, which can be abbreviated as **`wval`**.
  - Names interactively changes the width of the names region.

    The equivalent keyboard command is **`wnames`**, which can be abbreviated as **`wn`**.

Edit Off (Edit Menu)

The Edit Off command disables the editing mode initiated by the Edit → Insert, Edit → Replace, or Edit → Repeat commands.

The equivalent keyboard command is **`edoff`**, which can be abbreviated as **`edof`**.

Exit (File Menu)

The Exit command exits PROwave.

The equivalent keyboard command is **`quit`**, which can be abbreviated as **`q`**.

Full (View Menu)

The Full command displays the full view of the waveforms in the current simulation session.

The equivalent keyboard command is **`full`**, which can be abbreviated as **`f`**. You can also press the F4 key.

Grid (View Menu)

The Grid command controls the grid in the PROwave window. It has the following options:

- On turns the grid on.

  The equivalent keyboard command is **`gon`**.

- Off turns the grid off.

  The equivalent keyboard command is **`goff`**, which can be abbreviated as **`gof`**.

- Space changes the spacing of the grid. A prompt asks you for the horizontal step number.

  The equivalent keyboard command is **`gspace`**, which can be abbreviated as **`gs`**.

Help (Help Menu)

The Help command displays information about the selected command. See the "Design Entry" chapter for instructions on using this command.

Hostid (Help Menu)

The Hostid command displays the identification number of the security key found on the parallel port. This identification number also appears in the license file found in \proser\standard\license.wv.

Insert (Edit Menu)

The Insert command inserts the specified value (1, 0, X, Z) in a selected region of a digital waveform.

The equivalent keyboard command is **`insert`**, which can be abbreviated as **`ins`**.

Modify Signal (Edit Menu)

The Modify Signal command lets you adjust the signal display in PROwave. It has the following options.

- Shift shifts the transition times of the selected waveforms by the time that you specify. It creates a new waveform; the original waveform remains in the PROwave window, and the shifted waveform is automatically added to the status list.

  The equivalent keyboard command is **`shift`**, which can be abbreviated as **`sh`**.

- Scale scales the size of the selected waveform by the amount that you specify. It creates a new waveform; the original waveform remains in the PROwave window, and the scaled waveform is automatically added to the status list.

  The equivalent keyboard command is **`scale`**, which can be abbreviated as **`sca`**.

- Invert logically inverts values on a selected waveform.

  The equivalent keyboard command is **`invert`**, which can be abbreviated as **`i`**.

- And changes the logic of the selected signal to a logical AND. It creates a new waveform that appears in the PROwave window and on the status list.

  The equivalent keyboard command is **`and`**, which can be abbreviated as **`an`**.

- Or changes the logic of the selected signal to a logical OR. It creates a new waveform that appears in the PROwave window and on the status list.

  The equivalent keyboard command is **`or`**.

- Xor changes the logic of the selected signal to a logical XOR. It creates a new waveform that appears in the PROwave window and on the status list.

  The equivalent keyboard command is **`xor`**, which can be abbreviated as **`xo`**.

- Bus combines the selected signals into a logical bus. The new bus appears in the PROwave window and on the status list. The signal selected first is the most significant bit, and the signal selected last is the least significant bit.

  The equivalent keyboard command is **`bus`**, which can be abbreviated as **`bu`**.

- Expand breaks a bus into two or more signals that are added to the PROwave window and the status list.

  The equivalent keyboard command is **`expand`**, which can be abbreviated as **`ex`**.

New (File Menu)

The New command opens or merges a new generic output file. If the merged file is generic, the file in the window must also be generic.

The equivalent keyboard command is **`wgeneric`**, which can be abbreviated as **`wgen`**.

Open (File Menu)

The Open command opens a waveform file so you can view the waveforms produced by the simulation. It activates the Open dialog box; an example is shown Figure 11-1.

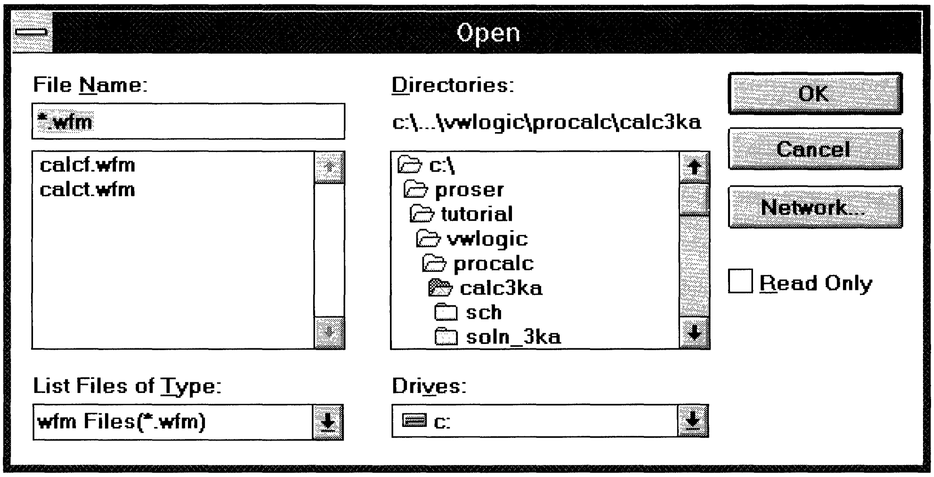


Figure 11-1 Open Dialog Box

The dialog box contains the following fields.

- File Name consists of a space for you to type in the name of the waveform file that you want to load (backspace over the asterisk) or a list box where you can click on the name of the design.
- List Files of Type indicates the type of file to be opened. The available types are WFM, DGT, SEE, and GEN files.
- Directories lists the directories for the specified drive so you can select the directory that contains the waveform file that you want to load.
- Drives lists the drives to which the PC is connected so you can select the drive on which the waveform file is located.
- Network accesses the network drive so you can connect to a printer on the network. It activates the Printers - Network Connections dialog box, shown under the Printer Setup command

in this chapter. Consult the Microsoft Windows documentation or click on the Help button for an explanation of the fields on this dialog box. The Network option is only available if you are running Windows for Workgroups.

- Read Only, if selected, prohibits writing to the currently loaded file.

The Open command has no equivalent keyboard command.

Pan (View Menu)

The Pan command establishes a specified point as the center of the window.

The equivalent keyboard command is **`pan`**. You can also press the F6 key.

Paste (Edit Menu)

The Paste command on the Edit menu inserts the contents of the clipboard at a specified point on a signal's waveform. The transitions of the original signal that are to the right of the insert point are shifted right by the length of the pasted waveforms region. All the spaces between the original and pasted transitions remain the same.

The equivalent keyboard command is **`rpaste`**, which can be abbreviated as **`rp`**.

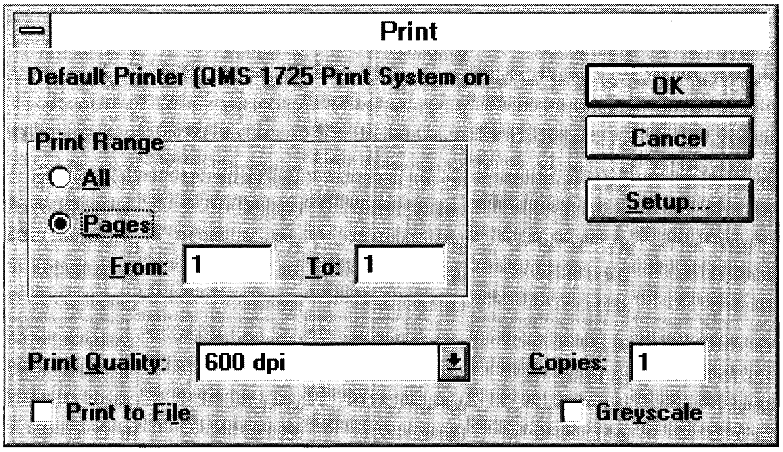
Paste (Change Menu)

The Paste command on the Change menu inserts the contents of the clipboard at the specified point in the signal.

The equivalent keyboard command is **`bpaste`**.

Print (File Menu)

The Print command prints the waveform in the active window. It activates the File Print dialog box, shown in Figure 11-2.



Default Printer (QMS 1725 Print System on

Print Range
○ All
◉ Pages
From: 1 To: 1

Print Quality: 600 dpi
Copies: 1
☐ Print to File
☐ Greyscale

Figure 11-2 File Print Dialog Box

This dialog box contains the following fields.

- Print Range determines whether to print all waveforms or only the range of waveforms that you specify.
- Print Quality controls how many dots per inch the printer will use when printing.
- Print to File places the waveform in a file that you can print.
- Setup brings up the Print Setup dialog box. Refer to your Microsoft Windows documentation for a description of the fields on this dialog box.
- Copies indicates how many copies of the waveform to print.
- Greyscale converts color on the screen to grayscale, which is a continuous tone image made up of a number of shades of gray.

PROhelp (Help Menu)

The PROhelp command displays online product information for the current PROwave release. You can obtain information on release notes, installation procedures, software problem reports, technical reference information, technical support services, and product data sheets and pricing.

Range (View Menu)

The Range command increases the magnification of the waveforms in the area that you specify in the active window.

The equivalent keyboard command is **`range`**, which can be abbreviated as **`ra`**. You can also press the F9 key.

Refresh (View Menu)

The Refresh command restores the view in the active window.

The equivalent keyboard command is **`refresh`**, which can be abbreviated as **`refr`**. You can also press the F5 key.

Remove (Waveform Menu)

The Remove command deletes the selected waveforms from the PROwave window.

The removed signals are not deleted from the status list and can be re-added to the window with the Waveform → Add command.

The equivalent keyboard command is **`remove`**, which can be abbreviated as **`r`**.

Rename (Waveform Menu)

The Rename command changes the names of selected signals.

The equivalent keyboard command is **`rename`**, which can be abbreviated as **`ren`**.

Repeat (Edit Menu)

The Repeat command copies a selected region of a digital waveform and repeats the region a specified number of occurrences to the right of the original.

The equivalent keyboard command is **`repeat`**, which can be abbreviated as **`repe`**.

Replace (Edit Menu)

The Replace command replaces the selected digital waveform region with a new value (0, 1, X, Z).

The equivalent keyboard command is **`replace`**, which can be abbreviated as **`repl`**.

Save (File Menu)

The Save command saves the waveforms in the active window in a PROwave trace file.

The equivalent keyboard command is **`save`**, which can be abbreviated as **`sa`**.

Save As (File Menu)

The Save As command saves the waveforms to another file. If the file already exists, PROwave prompts you to verify that you want to overwrite it. You can save the waveforms to one of the following types of files.

- Cmdfile saves the current signal data to a command file that PROsim can read.

  The equivalent keyboard command is **`cmdfile`**, which can be abbreviated as **`cmd`**.

- Generic saves the current signal data to a generic-format file.

  The equivalent keyboard command is **`ogeneric`**, which can be abbreviated as **`ogen`**.

- SaveFile saves the waveforms in a PROwave trace file.

  The equivalent keyboard command is **`save`**, which can be abbreviated as **`sa`**.

Set Radix (Waveform Menu)

The Set Radix command changes the base representation of the selected bus waveforms in the PROwave window. You can select any of the following radices.

- Binary Radix changes the display radix to binary.

  The equivalent keyboard command is **`rbin`**, which can be abbreviated as **`rb`**.

- Octal Radix changes the display radix to octal.

  The equivalent keyboard command is **`roct`**, which can be abbreviated as **`ro`**.

- Decimal Radix changes the display radix to decimal.

  The equivalent keyboard command is **`rdec`**, which can be abbreviated as **`rd`**.

- Hex Radix changes the display radix to hexadecimal.

  The equivalent keyboard command is **`rhex`**, which can be abbreviated as **`rh`**.

Signals (Waveform Menu)

The Signals command contains options that allow you to create, delete, and manipulate the specified signals in the PROwave window.

- FromSch creates a new PROwave signal with the same name as the signal selected in the PROcapture window.

  The equivalent keyboard command is **`fromsch`**, which can be abbreviated as **`fr`**.

- Create creates a new digital signal of a specified name and width in the active PROwave window.

  The equivalent keyboard command is **`create`**, which can be abbreviated as **`cre`**.

- Delete removes the selected signals from the waveform window.

  The equivalent keyboard command is **`delete`**, which can be abbreviated as **`del`**.

- Scroll places the selected signal at the top of the waveform window.

  The equivalent keyboard command is **`scroll`**, which can be abbreviated as **`scr`**.
- Select selects all signals at the same time.

  The equivalent keyboard command is **`selall`**, which can be abbreviated as **`sela`**.

Snap On (Change Menu)

The Snap On command turns on grid snapping, which anchors transitions to the nearest grid point. Grid snapping ensures that transitions are located at the proper place in the waveform area.

The equivalent keyboard command is **`son`**.

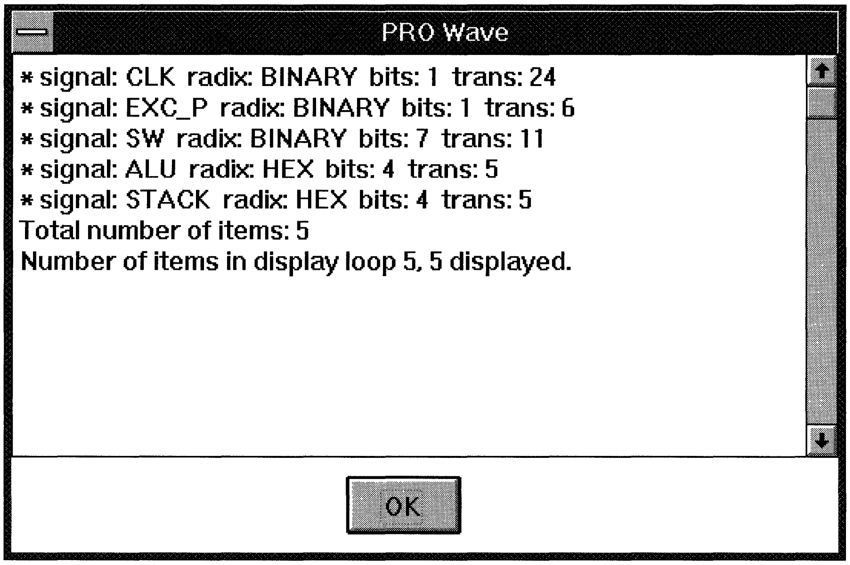
Snap Off (Change Menu)

The Snap Off command turns off grid snapping, which anchors transitions to the nearest grid point.

The equivalent keyboard command is **`soff`**, which can be abbreviated as **`sof`**.

Status (Info Menu)

The Status command displays a window containing information about the signals in the PROwave window. Figure 11-3 gives an example of this information window.



```
* signal: CLK radix: BINARY bits: 1 trans: 24
* signal: EXC_P radix: BINARY bits: 1 trans: 6
* signal: SW radix: BINARY bits: 7 trans: 11
* signal: ALU radix: HEX bits: 4 trans: 5
* signal: STACK radix: HEX bits: 4 trans: 5
Total number of items: 5
Number of items in display loop 5, 5 displayed.
```

Figure 11-3 Status Window

The equivalent keyboard command is **`status`**, which can be abbreviated as **`st`**.

Subst (Change Menu)

The Subst command substitutes the contents of the clipboard at a specified location on a signal.

The equivalent keyboard command is **`rsubst`**.

Tile (Window Menu)

The Tile command arranges the open windows in rows across the PROwave screen. The windows are fitted into all available screen space without overlapping. The "Copying Schematics" section of the "Design Entry" chapter gives an example of tiled windows.

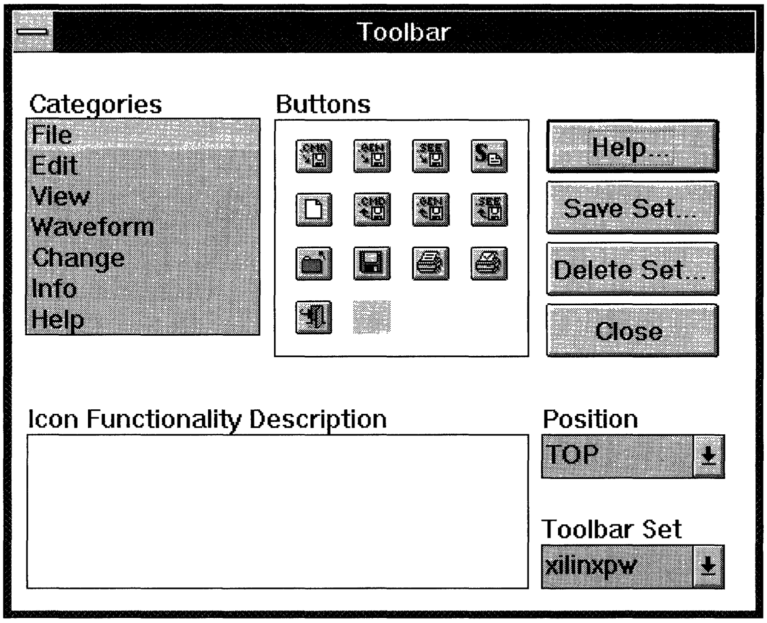
TLimit (Change Menu)

The TLimit command defines the time range in the PROwave window. Any transitions outside the region are deleted.

The equivalent keyboard command is **`tlimit`** *ltime rtime*, where *ltime* is the left time limit, and *rtime* is the right time limit.

Toolbar (View Menu)

The Toolbar command determines which icons are displayed in the toolbar of the PROwave window. It brings up the Toolbar dialog box, shown in Figure 11-4. See the "Toolbar Icons" section of this chapter for a description of most of the icons on the toolbar.



Position: TOP

Toolbar Set: xilinxpw

Figure 11-4 Toolbar Dialog Box

The following fields appear on this dialog box.

- Categories determines the type of icons that appear in the Buttons field. For example, if you select File, only the icons that govern files, such as the Print icon or the Open icon, appear in the Buttons box.
- Buttons displays the category of icon selected in the Categories list box. You can add an icon to the toolbar by clicking on the icon and dragging it to the toolbar. Similarly, you can delete an icon from the toolbar by dragging it into the Buttons field. When you click on an icon in this field, a brief description of its function appears in the Icon Functionality Description field.
- Save Set saves the configuration of icons in the toolbar that you arranged by dragging icons from the Buttons field.
- Delete Set removes the set of icons saved with the Save Set option.
- Close closes the dialog box without effecting any changes.
- Position governs the placement of the toolbar on the PROwave screen. You can place it on the top, bottom, left-hand side, or right-hand side of the screen.
- Toolbar Set loads the set of icons that you saved with the Save Set command.

TStep (Change Menu)

The TStep command changes the time step and/or the time units of all signals to a different order of magnitude. It changes the time axis but not the size of the waveforms.

You can change the time step to nanoseconds, picoseconds, microseconds, milliseconds, and seconds.

The equivalent keyboard command is **`tstep`** *new_tunit*.

Zoom In (View Menu)

The Zoom In command magnifies the visible area by a factor of two.

The equivalent keyboard command is **`in`**. You can also press the F7 key.

Zoom Out (View Menu)

The Zoom Out command shrinks the visible area by a factor of two.

The equivalent keyboard command is **`out`**. You can also press the F8 key.

Toolbar Icons

You can access some of the PROwave commands through toolbar icons. The PROwave window displays the following icons. When you move the cursor over each icon, a brief description of its function appears on the PROwave command line.

Open Icon

The Open icon, shown in Figure 11-5, opens a PROwave window so you can view the waveforms produced by the simulation. It activates the Open dialog box, an example of which is given in the "Open" section of this chapter. This command is equivalent to the File → Open command.

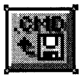


Figure 11-5 Open Icon

Close Icon

The Close icon, shown in Figure 11-6, closes the PROwave window opened with the File → Open command. It is equivalent to the File → Close command.

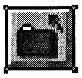


Figure 11-6 Close Icon

Save As Icon

The Save As icon, shown in Figure 11-7, saves the waveforms in a PROwave trace file. If the file already exists, PROwave prompts you to verify that you want to overwrite it. This command is equivalent to the File → Save As command.

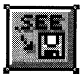


Figure 11-7 Save As Icon

New Icon

The New icon, shown in Figure 11-8, opens or merges a new generic output file. If the merged file is generic, the file in the window must also be generic. This command is equivalent to the File → New command.

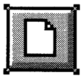


Figure 11-8 New Icon

Print Icon

The Print icon, shown in Figure 11-9, prints the waveform in the active window. It is equivalent to the File → Print command.

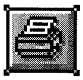


Figure 11-9 Print Icon

Exit Icon

The Exit icon, shown in Figure 11-10, exits PROwave. It is equivalent to the File → Exit command.

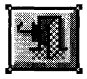


Figure 11-10 Exit Icon

Cut Icon

The Cut icon, shown in Figure 11-11, cuts the selected waveforms from the names region and places them in the clipboard, overwriting the previous contents of the clipboard. It is equivalent to the Change → Cut command.

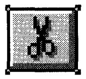


Figure 11-11 Cut Icon

Copy Icon

The Copy icon, shown in Figure 11-12, copies selected waveforms into the clipboard, overwriting the previous clipboard contents. It is equivalent to the Change → Copy command.

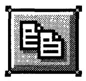


Figure 11-12 Copy Icon

Paste Icon

The Paste icon, shown in Figure 11-13, inserts the contents of the clipboard at the specified point in the signal. It is equivalent to the Change → Paste command.

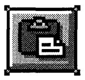


Figure 11-13 Paste Icon

Annotate Icon

The Annotate icon back-annotates simulation values in PROcapture and creates a back-annotation file called *project*.ann. This command writes to the ANN file each time that you move the crosshair. It is equivalent to the Change → Annotate On command.

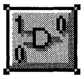


Figure 11-14 Annotate Icon

Color Icon

The Color icon, shown in Figure 11-15, changes the color of the selected waveform to the color that you specify. A color spectrum appears for you to select the new color. This command is equivalent to the Waveform → Display Params → Color command.

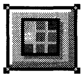


Figure 11-15 Color Icon

Schematic Signal Icon

The Schematic Signal icon, shown in Figure 11-16, creates a new PROwave signal with the same name as the signal selected in the PROcapture window. It is equivalent to the Waveform → Signals → FromSch command.

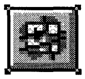


Figure 11-16 Schematic Signal Icon

Binary Radix Icon

The Binary Radix icon, shown in Figure 11-17, changes the display radix of selected bus signals or vectors to binary. It is equivalent to the Waveform → Set Radix → Binary Radix command.

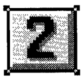


Figure 11-17 Binary Radix Icon

Octal Radix Icon

The Octal Radix icon, shown in Figure 11-18, changes the display radix of selected bus signals or vectors to octal. It is equivalent to the Waveform → Set Radix → Octal Radix command.

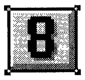


Figure 11-18 Octal Radix Icon

Decimal Radix Icon

The Decimal Radix icon, shown in Figure 11-19, changes the display radix of selected bus signals or vectors to decimal. It is equivalent to the Waveform → Set Radix → Decimal Radix command.

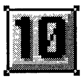


Figure 11-19 Decimal Radix Icon

Hexadecimal Radix Icon

The Hexadecimal Radix icon, shown in Figure 11-20, changes the display radix of selected bus signals or vectors to hexadecimal. It is equivalent to the Waveform → Set Radix → Hex Radix command.

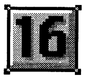


Figure 11-20 Hexadecimal Radix Icon

Help Icon

The Help icon, shown in Figure 11-21, displays information about the selected command. See the "Design Entry" chapter for instructions on using this icon. This command is equivalent to the Help → Help command.

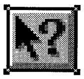


Figure 11-21 Help Icon

Viewlogic Interface Guide

Glossary

Appendix A

Glossary

This chapter defines the key terms and concepts that you need to understand to use the Viewlogic programs effectively.

Annotation

Annotation is the insertion of simulation values into the schematic.

Attribute

An attribute is an instruction placed on symbols or nets in a schematic to indicate its placement, implementation, naming, directionality, and so forth.

Back-Annotation

Back-annotation is the translation of a routed or fitted design to a timing simulation netlist.

Block

A block is a schematic or symbol sheet. There are four types of blocks:

- A Composite block indicates that the design is hierarchical.
- A Module block is a symbol with no underlying schematic.
- A Pin block represents a schematic pin.
- An Annotate block is a symbol without electrical connectivity that is used only for documentation and graphics.

Breakpoint

A breakpoint is a condition at which a simulator must stop to perform simulation commands.

Bus

A bus is a group of common signals. These signals are grouped together to facilitate manipulation on the schematic and viewing during simulation.

Command File

In simulation, a command file is a file containing a list of commands that assign vectors, generate input waveforms and clocks, and display signals. It is submitted for execution during simulation. You can create a command file with a text editor or from a set of input waveforms.

Component

A component is an instantiation or symbol reference from a library of logic elements that can be placed on a schematic.

Dangling Bus

A dangling bus is a bus connected to a component pin or net at one end and unconnected at the other. A small filled box at the end of the bus indicates a dangling bus.

Dangling Net

A dangling net is a net connected to a component pin or net at one end and unconnected at the other. A small filled box at the end of the net indicates a dangling net.

Label

A label is text attached to a bus, pin, net, or component that identifies it.

Net

A net is an electrical connection between components or nets. It can also be a connection from a single component. It is the same as a wire or a signal.

Netlist

A netlist is a text description of the circuit connectivity. It is a list of connectors, a list of instances, and, for each instance, a list of the signals connected to the instance pins. In addition, the netlist contains attribute information.

Node

A node is the junction of nets joined throughout the design hierarchy by pins on symbols.

Period

The period is the number of steps in a clock pattern multiplied by the step size.

Pin

A pin can be a package pin or a symbol pin. A package pin is a physical connector on an integrated circuit package that carries signals into and out of an integrated circuit. A symbol pin, also referred to as an instance pin, is the connection point of an instance to a net.

Radix

A radix is the base — usually binary, octal, decimal, or hexadecimal — in which waveforms are displayed in a waveform viewer such as PROwave.

Schematic

A schematic is a hierarchical drawing representing a design in terms of user and library components.

Sheet

A sheet is a page of a schematic.

Signal

A signal is a wire or a net. See "Net."

Simulation

Simulation is the verification of a design. Functional simulation verifies a design's logic, and timing simulation uses worst-case delays to perform verification.

Simulation Network

A simulation network is a file submitted to the simulator for functional or timing simulation.

Step

A step is the length of time that each value in a clock pattern is simulated.

Step Size

Step size is the length in nanoseconds of one step of a clock pattern.

Symbol

A symbol is a graphical representation of one level of hierarchy.

Tick

A tick is a simulation unit in a simulator such as PROsim or a waveform viewer such as PROwave. It can be a nanosecond, a millisecond, and so forth.

Vector

A vector is a group of signals that has been renamed for convenience during simulation. It is similar to a bus; "bus" refers to a group of signals on the schematic, and "vector" refers to a group of signals during simulation.

Watch List

A watch list is a list of nodes whose values are to be reported during simulation.

Waveform

A waveform is a graphical representation of a set of simulation transitions that depicts the digital or electrical values of a node on the schematic.

Wire

A wire is a net or a signal. See "Net."

Viewlogic Interface Guide

Program Options

Appendix B

Program Options

This appendix describes the options available in the translators that Viewlogic invokes to process your design.

Where you must supply parameters in addition to the option itself, the full syntax is shown below the option.

XNF2WIR Options

XNF2WIR offers several options, which are described in this section.

–b

This option turns off the continual output status normally issued by the program. When messages are written to a file, the status line becomes a single extremely long line that many text editors cannot process.

–c

–c *crs_filename*

Use this option to specify the name of the CRS file from your pre-routed schematics.

–l

This option places library aliases into the output WIR file.

–m

–m *maximum_number*

This option defines a maximum number of symbols to be placed in a single WIR file. It is extremely useful when XNF2WIR runs out of memory while processing your netlist. By breaking up the output into several different WIR files (.1, .2, .3 . . .), you can process very large netlists without running out of memory. The default value is 1000.

–r

The –r option creates an XMM file for simulating ROMs only.

–s

This option retrieves the OSC parameterized attributes that were set in ViewDraw and places them in the output WIR subdirectory file. This option is only needed if the OSC symbol is used in the design.

Remember to specify the CRS file with the –c option.

–x

The CRS file was created to preserve the OSC parameterized attributes and hierarchical net names that were changed by WIR2XNF in the process of translating the WIR subdirectory file to an XNF file. This option returns the net names to their original format.

Remember to specify the CRS file with the –c option.

VSM Options

The following options are available in VSM.

–d

–**d***tablename*

The –d option specifies the name of a delay back-annotation table file with the default extension of .dbt. See the Viewlogic documentation for a description of the back-annotation files. In the command line syntax, *tablename* is the name of the back-annotation table file. There is no space between –d and the table name.

–f

*–***f***filename*

The –f option specifies a name for the output VSM file other than the default name of *project*.vsm. In the command line syntax, *filename* is the name of the output VSM file. There is no space between –f and the file name.

–h

The –h option generates full hierarchical net name equivalents in the wirelist, allowing you to observe simulation values of logically equivalent nets at any level in the design hierarchy. This option is the default.

–s

The –s option references all nets by their top-level net names as it appears in the design. It reduces the size of the wirelist file by a factor of two. This option halves the memory requirements of ViewSim, effectively doubling the simulator's capacity.

–t

The –t option includes additional records for fault simulation in the VSM file for Viewlogic ViewFault. These records identify pins and allow all nodes and pins at or below a given level to be analyzed for faults.

–w

The –w option generates a description of the design with flattened connectivity. It is useful for ASIC designs, where post-layout delay information is back-annotated to the design. You can then instantiate the design on a larger circuit for system-level simulation. You must move the output file to the appropriate wir subdirectory so that it is visible to the ensuing wirelist execution.

VSMUPD Options

VSMUPD offers three options.

–b

This option turns off the continual output status normally issued by the program. When messages are written to a file, the status line becomes a single extremely long line that many text editors cannot process.

–o

–o *output_filename*

When this option is specified, VSMUPD creates the specified output VSM file. If you do not specify this option, the output file overwrites the routed VSM file.

–x

–x *input_filename*

This option specifies the post-routed XNF file name when it differs from the post-routed WIR file name. They differ only when you use XNF2WIR to change the name of the post-routed design. This change is not standard procedure; normally, you would change the name during the LCA2XNF conversion.

VMH2XNF Options

The following options are available in VMH2XNF.

–l 4, –l 5

The –l 4 or –l 5 option allows you to choose the V4.0 or V5.0 Xilinx EPLD library, regardless of the library you used to create your design. The default is the V5.0 library for a new design or the V4.0 library for a design created with V4.0 components. You may want to use the –l 4 option if you are using the XNF file with third-party software that has not yet been upgraded to V5.0, for example, logic modeling software.

–o

The –o option changes the name of the output file, minus the extension. It prevents your timing simulation XNF file from overwriting the XNF file that contains your design.

FITNET Options

The following options are available in FITNET.

–i

The –i option ignores the pin assignments in the schematic.

–u

The –u option drives unused I/O pads, clock signals, FOE signals, and CE pins to GND.

ViewGen Options

ViewGen command line options follow the input and output file name. You can store a set of options in a file and include them with the –c option. For example, a file called viewgen.cmd may contain the following ViewGen commands.

```
-makesym
-inpin VCC GND
-symattr LEVEL=STD
```

If the VIEWGENOPT environment variable is set in the viewgen.ini file, it is assumed to be the name of an options file that is automatically included, and it is processed before the command line options. If VIEWGENOPT is not specified, the system attempts to read viewgen.ini as a command file in the current directory or in the wdir directory.

Options read from a file can be separated by either a space or a return character. You can use multiple –c options, which are processed in the order of occurrence. Except for the –c option, if an option is specified more than once, its last occurrence overrides all others.

ViewGen also reads the viewdraw.ini file from the current directory or a directory in a path list specified by the WDIR environment variable. ViewGen uses the DIR , ADISTANCE, GRID, NETNAME, and TEXTSIZE parameters from the viewdraw.ini file.

- The DIR parameter finds the input WIR file and all symbols referenced in the input file and border page symbols.
- The ADISTANCE and GRID parameters supply default values for ViewGen command options.
- The TEXTSIZE and NETNAME parameters specify the size of text labels and the names of nets that are to be made global.

See the Viewlogic documentation for further descriptions of these parameters.

Note: It is strongly recommended that the –Sheet option be used for large designs. This option creates several sheets of the indicated sheet size. If this option is not used, you may receive DOS/16M unexpected-interrupt errors or protected-mode errors.

Common Options

The more commonly used ViewGen options are the following.

–c

–c *optionfile*

The –c option specifies a file containing all command line options to be read. The file is formatted just like the command line, except a new option can begin on a new line in the file. You can specify more than one –c option; this option can be nested inside a file.

–help

The –Help option prints a summary of the command line options.

–i

–i [*lib*:] *infile*

The –i option specifies the input file as an option inside a command file. When specifying the input file, specify only its name, not the full directory path; the viewdraw.ini file specifies the directories to be searched.

–makesym

Use the –Makesym option to automatically generate the top-level symbol when you run ViewGen. By default, the program does not automatically create the symbol.

–noschem

Using the –Noschem option, you can instruct ViewGen not to create a schematic. This option is usually used in conjunction with the –Makesym option, in which ViewGen is used as a symbol generator.

–o

–o [*lib*:] *outfile*

The –o option specifies the output file as an option inside a command file. If the last character in the output file is "/" or "\", it is assumed to be the output file directory, and the name of the output file will match the input file name.

–quiet

Using the –Quiet option, you can suppress the display of informational messages while the program is running.

–sheet

–**sheet** *size*

The –Sheet option specifies the sheet size for page splitting and framing. *Size* is one of the following parameters.

AL	11≤x 8.5≤
BL	17≤x 11≤
CL	22≤x 17≤
DL	34≤x 22≤
EL	44≤x 34≤
AP - P	Same as AL-EL, rotated for portrait
A4 - A0	European standard sizes
A4P - A0P	European portrait sizes

This option forces ViewGen to fit the design into the specified sheet size. If the design cannot fit into a single sheet, it is split into multiple sheets of the specified size.

When multiple sheets are generated, one or more sheets may overflow the boundaries of the specified sheet size. ViewGen attempts to correct this problem by restarting itself with smaller height and level limits. See the descriptions of –Heightlimit and –Levellimit in the next section, "Advanced Options."

The number of times ViewGen restarts can be controlled with the –Restart option, described in the "Advanced Options" section, following.

If the –Sheet option is not specified, no pages are split, and the sheet size is adjusted to fit the single resulting schematic.

This option is recommended for large designs.

–sheet

`–sheet` *width height*

The –Sheet option can also specify a non-standard sheet size for page splitting and framing, with width and height specified in 0.01-inch units. This option overrides the –Sheet option.

If either the width or height of the page is specified as 0, the page size is adjusted to fit the resulting schematic. You can use this option to create a variable-size sheet with a fixed height or width.

–sheetsym

`–sheetsym` *size symbol*

The –Sheetsym option specifies the name of the border symbol for the specified sheet size. The symbol should be constructed as described in the "Sheet Symbols" section in the "Manual Translation" chapter.

For a non-standard sheet size, use Z as the size. The search path specified in the viewdraw.ini file is used to search for the symbol.

There can be a separate –Sheetsym option for each sheet size.

Included with ViewGen are default border symbols for all U.S. sheet sizes. The default viewgen.ini file provided with ViewGen contains –Sheetsym options specifying border symbols for all U.S. standard sizes.

Advanced Options

This section lists ViewGen's advanced options.

–busname

–**busname** *name*

Using the –Busname option, you can specify *name* so that all nets with *name* as a prefix and a number as a suffix are combined into a single bus and routed as such. For example, if you specify –busname A, all nets in the form A0, A1, ... A*n* are combined into a bus named A[*n*:0]. Multiple –Busname options are appended.

–compcomp

–**compcomp** *n*

The –Compcomp option sets the minimum spacing between components in 0.01-inch units. The default value is the ADISTANCE value read from the viewdraw.ini file; if ADISTANCE is not specified, the default value is 20.

–compnet

–**compnet** *n*

The –Compnet option sets the minimum spacing between components and nets in 0.01-inch units. The default value is the ADISTANCE value read from the viewdraw.ini file. Otherwise, the default value is the average of the component-to-component and net-to-net spacing. If neither of these are specified, the default value is 20.

–comptext

–**comptext** *n*

The –Comptext option sets the minimum spacing between components and text labels in 0.01-inch units.

–flatten

The –Flatten option specifies that the design be flattened all the way down to the primitive level. By default, ViewGen generates a schematic down to only one level in the hierarchy. If you use the

–Loadlevel option with the –Flatten option, the value specified by the –Loadlevel option overrides it.

–gridsize

–gridsize *n*

Use the –Gridsize option to set the size of the grid used for placing components, nets, and text on the sheet. If all symbol bounding boxes, pin locations, and minimum spacing distances are multiples of this grid distance, all objects in the drawing are placed on the grid. The default value is the GRID value read from the viewdraw.ini file; if GRID is not specified, the default value is 10.

–heightlimit

–heightlimit *n*

The –Heightlimit option specifies the maximum height in 0.01-inch units for component columns on a page, excluding the space occupied for nets. If not specified, a default value is implied from the sheet size. You can use this option to control page splitting and to correct sheet size overflow.

–helpadv

The –Helpadv option displays the advanced command options.

–inpin

-inpin *string*

Use the **–**Inpin option to specify the name of a string to be used as a name of an input I/O pin on the top-level symbol. You can use this option to place pins on a generated symbol. ViewGen also uses names specified by this option as a final means of finding I/O nets on the schematic. Multiple **–**Inpin options are appended.

–latchfblevel

-latchfblevel *n*

The **-**Latchfblevel option, like –Outfblevel, limits the number of levels to the right of the output connector column for certain components. If specified, the components with symbols identified as latch symbols have a maximum level of *n*. You can use this option to move the latched state machine outputs to the right side of the page, allowing the outputs to feed back to inputs on the left.

–levellimit

`-levellimit` *n*

The –Levellimit option specifies the maximum number of logic levels per schematic page, that is, the number of columns including input/ output connectors. If not specified, a default is implied from the sheet size. This option controls page splitting and corrects sheet-size overflow.

–loadlevel

`-loadlevel` *string*

The –Loadlevel option specifies a LEVEL attribute value, represented by *string*, to which the netlist should be flattened. By default, ViewGen only generates a schematic down to one level in the hierarchy.

–netnet

`-netnet` *n*

The –Netnet option sets the minimum spacing between nets in 0.01-inch units. The default value is the ADISTANCE value read from the viewdraw.ini file; if ADISTANCE is not specified, the default value is 20.

–outfblevel

`-outfblevel` *n*

The –Outfblevel option controls the level assignment algorithm, which divides the schematic into columns.

Normally, the signal flow is from left to right, so a component with an output pin is placed to the left of input pin components. If a component with an output pin is tied to a net with an output connector, you can limit the level number, counted from the rightmost column of output connectors, with this option. If the level of a component is limited, the net may feed back to inputs in a column to the left.

The default value is 2. You can generate a shorter and wider schematic by increasing the value of this option. If *n* is 2, components with an output connected to an external output or off-page connector are placed in the first or second column to the left of the output connectors. If *n* is 4, the components are placed in the first through fourth column to the left of the output connectors.

–outpin

`-outpin` *string*

Use the –Outpin option to specify the name of a string to be used as a name of an output I/O pin on the top-level symbol. You can use this option to place pins on a generated symbol. ViewGen also uses names specified by this option as a final means of finding I/O nets on the schematic. Multiple –Outpin options are appended.

–permute

`-permute` *symbol, permute...*

The –Permute option specifies which symbol pins can be permuted, or switched, when routing nets. Following this option is a list of permutability definitions, where each definition is a symbol name followed by a permutability definition string.

The *permute* string consists of one or more permutable pin groups, separated by commas. Each permutable pin group is a list of pin names, separated by commas and enclosed in parentheses. Instead of a single pin name, a group can include another subgroup enclosed in parentheses.

If the pins in a subgroup are not permutable and if the group can be permuted with another matching non-permutable group, the non-permutable groups are enclosed in brackets. If a permutable group contains a subgroup, each subgroup must have a matching structure. Multiple –Permute option definitions are appended.

Here are some examples.

- ((A,B),(C,D)) — A and B can be swapped and C and D can be swapped; as a group, A and B can be swapped with C and D.
- (A,B),(C,D) — A and B can be swapped and C and D can be swapped; as a group, A and B cannot be swapped with C and D.

- ([A,B],[C,D]) — A and B cannot be swapped and C and D cannot be swapped; as a group, A and B can be swapped with C and D.

–permuteinputs

`-permuteinputs` *symbol*

The –Permuteinputs option specifies a list of symbols with permutable input pins. To improve the appearance of the schematic, ViewGen may swap connections between inputs within these symbols. Multiple –Permuteinputs option definitions are appended.

–pinlblsize

`-pinlblsize` *n*

You can use the –Pinlblsize option to specify the size, in 0.1-inch units, of pin labels placed on ViewGen-generated symbols.

–restart

`-restart` *n*

Use the –Restart option to specify the number of times that the program can restart when the schematic is larger than the specified sheet size. If *n* is greater than 0, a size overflow restarts the program with reduced –Levellimit or –Heightlimit values. Setting *n* to 0 inhibits restart, prints a warning with recommended parameters, and generates a schematic with a larger sheet size. The default value is 2.

–symatt

`-symatt` *name=value*

Use the –Symatt option to add the specified attributes and values to the top-level symbol created with the –Makesym option. If the specified string contains spaces, it should be enclosed in double quotation marks. Multiple –Symatt options are appended.

–symattsize

`-symattsize` *n*

The –Symattsize option specifies the size, in 0.1-inch units, of attribute text placed on ViewGen-generated symbols.

–sympinlth

`-sympinlth` *n*

You can use the –Sympinlth option to specify the length, in 0.1-inch units, of pins on ViewGen-generated symbols.

–sympinspc

`-sympinspc` *n*

You can use the –Sympinspc option to specify the spacing, in 0.1-inch units, between pins on ViewGen-generated symbols.

Other Options

ViewGen's remaining options are listed in this section.

–analyze

When you use the –Analyze option, ViewGen prints a list of netlist errors.

–complabels

The –Complabels option adds labels to all components using the internal name if a component label is not supplied.

–deletenet

`-deletenet` *name...*

The –Deletenet option deletes the specified nets from the drawing without routing them. It is useful for removing the GR net in an XC2000 or XC3000 WIR file.

–labelpos

`-labelpos` *pos just*

Using the –Labelpos option, you can specify the default label text origin and its justification relative to the symbol bounding box. Position and justification values must be between 1 and 9, as explained in the following table.

Table D-1 Position and Justification Values

Value	Position
1	Upper left corner
2	Left center
3	Lower left corner
4	Top center
5	Center of box
6	Lower center
7	Upper right corner
8	Right center
9	Lower right corner

–latchsym

`-latchsym` *symbol1 symbol2*

Use the -Latchsym option to identify the symbols considered for the -Latchfblevel option. The levels of components with a matching symbol name are limited by the -Latchfblevel value. Multiple –Latchsym options are appended.

–maxsheet

`–maxsheet` *size*

The –Maxsheet option specifies the maximum sheet size that the system can use when generating a single-page drawing. *Size* is one of the standard sizes specified in the –Sheet option description. This option only works for single-sheet drawings and is ignored if the –Sheet option is specified.

–minsheet

`–minsheet` *size*

You can use the –Minsheet option to specify the minimum sheet size that the system can use when generating a single-page drawing. *Size* is one of the standard sheet sizes specified in the –Sheet option description. This option only works for single-sheet drawings and is ignored if the –Sheet option is specified.

–visattr

–visattr *name name*

You can use the –Visattr option to specify one or more names of component attributes that are to be made visible. You can use it to make an invisible attribute definition visible. When attributes are attached to components, the placement and visibility are defined by a matching attribute in the symbol definition. Multiple –Visattr options are appended. The names can include the asterisk (*) character for use as a wildcard.

WIR2XNF Options

You can modify the operation of the WIR2XNF program by using the following options.

–p

–p *parttype* [*-speedgrade*]

The -p option specifies the FPGA part type for your design if it is not already specified in the schematic design. You can also use this option to override the part type specified in the design file. The part type consists of two parts: the first part is the part name and package type, and the second part is the optional speed grade. Following is an example:

```
wir2xnf -p 4005PG156-5 [alphadesign]
```

WIR2XNF does not run unless you specify a part type, either with the –p option or by placing a PART attribute in the schematic.

–b

The –b option suppresses the output messages that WIR2XNF normally outputs to the screen to indicate its processing status.

–od

–od *subdirectory*

The –Od option specifies the path name of the directory in which to place the output XNF files.

Viewlogic Interface Guide

Warning and Error Messages

Appendix C

Warning and Error Messages

This appendix lists the warning and error messages that you may receive from the translators that Viewlogic invokes to process your design.

XNF2WIR

This section lists the warning and error messages that XNF2WIR produces.

Warning Messages

XNF2WIR may issue the following warning messages.

`Warning 11. Input ignored:` *input*`.`

Characters were found on the end of a record line after all expected data was received.

`Warning 19. Pin name` *pin_name* `used multiple times on symbol` *symbol_name*`.`

This message is issued when the given pin name is used more than once on a symbol. Rename the pins to correct this situation.

`Warning 30. Unknown MAP symbol type` *symbol_type* `for SYM` *symbol_name*`. Will use default MAP type` *map_type*`.`

This message is issued if the CLBMAP symbol has a MAP attribute value that is not PUC, PUO, PLC, or PLO. See the "XNF-MAP" chapter of the *Development System Reference Guide* for more information about MAP attributes. XNFMAP defaults to MAP=PUC.

`Warning 30.` *'Type' 'name'* `has invalid parameter value` *parameter*`.`

This message is issued if a floating-point value of the THI and TLO parameters have trailing characters that are not "ns" for nanoseconds. Only the "ns" suffix can be added to a THI or TLO value.

`Warning 30. Unknown SYM record parameter` *parameter* `ignored.`

This message is issued if an unknown XNF symbol parameter is found. Check the parameter specification in the design file.

`Warning 30.` *Parameter* `parameter will be ignored on SYM` *symbol_name*`, type` *symbol_type*`.`

This message is issued if a parameter is assigned to a symbol that does not support the parameter, such as a FAST attribute on a DFF symbol. Check the parameter for that symbol.

`Warning 30. Unknown PIN record parameter` *parameter* `ignored.`

This message is issued if an invalid parameter is assigned to a PIN. Check the parameter for that symbol pin.

`Warning 30. PIN record parameter` *parameter* `ignored on MAP symbol.`

This message is issued if an invalid parameter is assigned to a PIN statement for a CLBMAP. Only a P (Pin-Lock) net attribute is allowed on CLBMAP symbol pins.

`Warning 30. Unknown SIG record parameter` *parameter* `ignored.`

This message is issued if an invalid parameter is assigned to a SIG record. Check the parameter for that signal.

`Warning 30. Unknown EXT record parameter` *parameter* `ignored.`

This message is issued if an invalid parameter is assigned to a EXT record. Check the parameter for that I/O pad.

`Warning 30. FILE parameter will be ignored on` *symbol_name* `SYM, type` *symbol_type*`. (Non-flattened design?)`

This message is issued if a FILE attribute is found on a non-macro symbol. FILE attributes should be added only to unflattened macro symbols. This warning may occur if a reserved name, such as AND or OR, is used as the name of a FILE macro.

`Warning 30. Extra LOC parameter` *parameter* `found on signal` *signal_name* `will be ignored.`

This message is issued if more than one LOC constraint is found on a record. Multiple-block LOC constraints should be separated by semicolons (;). If commas (,) are used to separate LOC constraints, this warning is issued.

`Warning 31. Invalid LOC parameter` *parameter* `on symbol type` *symbol_name*`.`

An invalid location specification was found on the indicated symbol. Check the legal LOC constraints for the FPGA family in use.

`Warning 231. Unknown flag` *option_name* `ignored.`

The option used in the command line is not valid and is ignored. Re-enter the command line entry with a legal option. For a listing of legal options, see the "Program Options" chapter in this manual.

`Warning 232. Extra argument` *argument_name* `ignored.`

There is an extra argument specified at the command line; it is ignored. Unless the indicated argument is required, no action is required.

`Warning 233. Could not open CRS file` *filename* `for read.`

The specified cross-reference (CRS) file does not exist in the present directory. Make sure that the cross-reference file generated by the WIR2XNF program has the same name as the back-annotated XNF file and that both files are located in the same directory. You must rename the CRS file to match the routed XNF file.

`Warning 236. OSC pulse information line is missing PULSEHI and PULSELO data.`

An OSC information line in the cross-reference (CRS) file has been corrupted or changed. Regenerate the cross-reference file by re-running the WIR2XNF software.

`Warning 237. OSC pulse information line missing PULSELO data.`

An OSC information line in the cross-reference (CRS) file has been corrupted or changed. Regenerate the cross-reference file by rerunning the WIR2XNF software.

`Warning 239. More than one oscillator exists. PULSEHI and PULSELO set to default.`

Pulse information for an OSC primitive was found in the cross-reference (CRS) file, but an OSC component with the indicated instance name was not found in the design; there were also multiple OSC components in the design. Force the node normally driven by the oscillator to the proper levels during simulation by using the Wfm, Clock, H, and L commands in PROsim.

`Warning 243. Invalid format [`*format*`] within CRS file` *filename*.

There is a line in the CRS file that does not follow the CRS file format. Create a new CRS file using WIR2XNF on the top-level design. You should specify the output file name when running WIR2XNF so the original XNF file is not overwritten. Remember to rename the CRS file to match the input file name to XNF2WIR, then rerun.

`Warning 247. New INV symbol` *symbol_name* `created for pin` *pin_name* `of symbol` *symbol_name*.

`Warning 248. Attempt to create new INV symbol for PIN` *pin_name* `of SYMBOL` *symbol_name* `unsuccessful.`

This message is produced when a ViewBase routine fails to insert a necessary inverter in the WIR file. When translating your design, XNF2WIR encountered a component that required the insertion of an inversion symbol. For example, a clock signal that was inverted in a CLB must be represented in the WIR file by an inverter for simulation. It does not alter any timing parameters. This warning is a program error that should be reported to Xilinx Technical Support.

Error Messages

XNF2WIR may issue the following error messages.

```
Error 1. Field too long.
```

This message is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 2. Unexpected LCANET record.
```

This error indicates an LCANET record on any line other than the first.

```
Error 3. ENDMOD with no matching MODEL record.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 4. Symbol sub-record outside of symbol
group. Record ignored.
```

This message is issued when the XNF syntax is violated, and when records that should be within other records, that is, MODEL, ENDMOD, or PIN, are found outside the group.

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 5. Illegal record inside symbol group.
Record ignored.
```

This message is issued when XNF syntax is violated and when a record that is not allowed inside other symbol groups is found within a group, for example, a PROG record inside a SYM group.

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 6. Premature EOF record in symbol group.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 7. Illegal record inside MODEL group.
Record ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 8. Premature EOF record in MODEL group.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 9. Premature end-of-file. No EOF record
found.
```

An EOF record must conclude every XNF file. This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 10. Unknown record type` *type*`. Record ignored.`

XNF2WIR encountered an invalid XNF record type in the file. This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 15. Valid part type must be specified
before netlist symbols can be read.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 16. Invalid part record, missing part type.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 17. Missing name on SYM record. SYM record
ignored.
```

```
Error 17. Missing type on SYM record. SYM record
ignored.
```

`Error 17. Unknown symbol type` *type*`. SYM record ignored.`

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 18. Invalid PIN record. Missing name field. PIN record ignored.`

`Error 18. Invalid PIN record. Missing or invalid direction field. PIN record ignored.`

`Error 18. Invalid PIN record. Invalid direction field. PIN record ignored.`

`Error 18. Invalid PIN record. Non-numeric delay field. PIN record ignored.`

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 20. Missing command on CFG record. CFG record ignored.`

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 21. CFG records allowed only in CLB and IOB symbols. CFG record ignored.`

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 22. Invalid SIG record. Missing signal name. SIG record ignored.`

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 23. Invalid EXT record. Invalid direction` *value*`. EXT record ignored.`

EXT records must have a direction field of I, O, T, B, or U.

```
Error 23. Invalid EXT record. Missing signal
name. EXT record ignored.
```

```
Error 23. Invalid EXT record. Bad or missing
direction field. EXT record ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 24. Invalid BUS record. Missing bus name.
BUS record ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 25. Invalid PULSE record. Missing pin name
field.  PULSE record ignored.
```

```
Error 25. Invalid PULSE record. Missing or
invalid polarity field. PULSE record ignored.
```

```
Error 25. Invalid PULSE record. Invalid polarity
field.  PULSE record ignored.
```

```
Error 25. Invalid PULSE record. Invalid or
missing minimum width field. PULSE record
ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 26. Invalid PWR record. Bad or missing
polarity field. PWR record ignored.
```

PWR record must have a 1 or a 0 in the polarity field.

```
Error 26. Invalid PWR record. Polarity is value.
Must be '0' or '1'. PWR record ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

```
Error 27. Invalid SETUP record. Missing pin name
field.  SETUP record ignored.
```

```
Error 27. Invalid SETUP record. Missing clock pin
name field. SETUP record ignored.
```

```
Error 27. Invalid SETUP record. Missing or
invalid clock edge field. SETUP record ignored.
```

```
Error 27. Invalid SETUP record. Invalid clock
edge field.  SETUP record ignored.
```

```
Error 27. Invalid SETUP record. Missing or
invalid setup time. SETUP record ignored.
```

```
Error 27. Invalid SETUP record. Missing or
invalid hold time. SETUP record ignored.
```

This error is an indication of an invalid or corrupted XNF file. Rerun the program that created this XNF file and try again. If the error persists, check with the company that supplied the product.

`Error 53. Out of memory. Needed` *number* `bytes.`

There is insufficient memory to complete the XNF file processing. Check the memory requirements for the FPGA part type in use.

```
Error 200. Can only process files with the .xnf
suffix. (xnf default)
```

XNF2WIR only accepts XNF files as its input. Verify that the input file name is correct and that an XNF file with the specified name exists. If there is no XNF file in the current directory, you must create one using LCA2XNF. Adding the suffix is not necessary when specifying the file name, because the default is XNF.

`Error 201. Unable to open` *filename* `for` *function*`.`

The indicated file could not be read from or written to. Check to be sure that your hard disk drive is not full and that you spelled the file name correctly.

```
Error 202. Output name greater than eight
characters long.
```

`Error 209. Group` *group_name* `could not be created.`

The top-level project group could not be created because of an invalid project name or insufficient memory. Rename the project or add more memory to your system.

`Error 210. Could not load symbol` *symbol_name*`.`

The indicated symbol could not be found using the current directory search paths defined in your viewdraw.ini file. Place the appropriate directory search path in your viewdraw.ini file.

`Error 214. XNF pin names for` *component* `do not match VIEWLOGIC symbol.`

The pin names in the specified symbol do not match the names on the Viewlogic symbol.

`Error 217. The logical function of` *s* `and its Viewlogic model do not match. Please check to ensure the SCHNM attribute is either properly specified or absent.`

`Error 218. Internal error, cannot open temporary file` *filename*`.`

`Error 220. Internal error, file` *filename* `not complete. Possible system error.`

`Error 221. Out of memory.`

`Error 222. No viewdraw.ini file found in current directory.`

VSM

This section lists the warning and error messages for VSM.

Warning Messages

VSM may issue the following warning messages.

`Warning: No component or pin instances were read from file` *filename*`. Make sure the file contains the token .BA to begin reading.`

This warning indicates that something may be wrong with the back-annotation file format. The .BA token synchronizes VSM with the file.

`Warning: Unrecognized record` *record* `in back-annotation file.`

This warning indicates that something may be wrong with the back-annotation file format. The supported record types are C, P, and A. In addition, you can use the comment bar " | " to embed readable text into the table.

`Warning: Duplicate instance` *instance_name* `in back-annotation file.`

The back-annotation file can contain only one instance of any given component or pin.

`Warning: Instance` *instance_name* `found in back-annotation file not referenced by design.`

Most likely, the tool used to create the back-annotation file did not generate instance names that correspond to the PROcapture-generated instance names. Check that the design has not been modified since the back-annotation file was created.

`Warning: Missing PINTYPE attribute for pin` *pin_name* `of symbol` *symbol_name*`.`

Library cell pins must have PINTYPE attributes on them.

`Warning: Back-annotation to output or bidirectional pin` *pin_name* `detected.`

Delay back-annotation is only supported to input (PINTYPE=in) pins.

`Warning: Cannot back-annotate delays to pins at multiple levels within the same pin hierarchy. Delay back-annotated to lower pin` *pin_name* `used. Delay back-annotated to upper pin` *pin_name* `ignored.`

VSM has encountered the same logical pin instance two or more times when resolving the back-annotated delay. It will use the delay associated with the lowest-level pin.

`Warning: Back-annotated pins on net` *net_name* `have multiple drivers.`

A back-annotated pin is being driven by more than one gate. The back-annotated delay will be applied from all outputs to the input.

Error Messages

VSM may issue the following error messages.

`Error: Cannot open VSM (or WIR) file:` *filename*`. Aborting output.`

The operating system open file routine is failing. Check that the file allocation parameter is a reasonable number, that is, a number greater than 5. Another possibility is that your operating system limits the number of files per directory and you have exceeded the limit; delete some unneeded files and execute the program again. Refer to your operating system documentation for further information about file allocation.

`Error: Could not open back-annotation file` *filename*`.`

The operating system open file routine is failing again. This time, however, it presumes the file already exists and is opening it for read access. Make sure the file is in the directory from which you are running the program or is available from the WDIR environment variable.

`Error: Back-annotation index and file` *filename* `inconsistent.`

This message indicates that the back-annotation text file is corrupt, possibly because it was created on another machine. Create the text file on the machine that you will use to run the netlist program. Alternatively, you can rewrite the file on the local platform.

`Error: Back-annotated input pin` *pin_name* `is connected to an output pin` *pin_name* `and an input pin` *pin_name* `within the same cell.`

This error message indicates a violation of the restriction that no cell input pins be connected to module input and output drivers within the cell.

VSMUPD

This section lists the warning and error messages for VSMUPD.

Warning Messages

VSMUPD may issue the following warning messages.

`Warning 2. Unknown flag` *flag_name* `ignored.`

An option specified on the command line is not one of the valid options for VSMUPD. For a list of valid options, see the "Program Options" chapter in this manual, or enter **vsmupd** to see a list of options and their input format.

`Warning 3. Extra argument` *argument* `ignored.`

An option specified on the command line was not needed. For a list of valid options and their arguments, see the "Program Options" chapter in this manual, or enter **vsmupd** to see a list of options and their input format.

```
Warning 4. Missing file name for -o option.
Option ignored.
```

When the –o option is specified, the program automatically creates a new VSM file with the specified file name output. If you do not specify the output file name with this option, it is ignored. For the correct input format, see the "Program Options" chapter in this manual, or enter **vsmupd** to see a list of options and their input format.

Error Messages

VSMUPD may issue the following error messages.

```
Error 1. Can only process files with the .vsm
suffix. (vsm default)
```

VSMUPD only processes files that have the .vsm suffix. Check the input file name and verify that there is a VSM file with the specified name. If the VSM file does not exist, run VSM on the routed design, then rerun VSMUPD. The suffix does not need to be specified because the default is .vsm.

`Error 2. Unable to open file` *filename* `for` *function*`.`

The specified file could not be opened for the indicated function. This message indicates that the file does not exist, was specified incorrectly, or contains attributes that prevent the file from being read from or written to. Check the file for any of these possible conditions and rerun VSMUPD.

`Error 3. Could not` *function* `file` *filename*`.`

The file could not be read from or written to because no file exists or the file contains attributes that prevent it from being written to or read from. Check the file for any of these possible issues and rerun VSMUPD.

ViewGen

Following are the ViewGen error messages.

Error Messages

```
ll nets are non $; A schematic symbol definition
is required.
```

ViewGen cannot determine the external nets that have input or output connectors on the schematic. The input netlist does not have pin definitions, and a symbol for the schematic is not available. The last recourse pin definitions using nets labeled as external do not work because all nets were labeled.

To correct this problem, define a symbol for the schematic in the parallel sym subdirectory from the wir and sch subdirectory containing the pin definitions that specify the external nets in the schematic.

```
Border symbol name does not have LOWER_LEFT and
UPPER_RIGHT.
Border symbol name does not have valid LOWER_LEFT
and UPPER_RIGHT.
```

A border symbol placed on the sheet that contains a frame for the sheet must define the rectangle in which the schematic is placed. This rectangle is defined by placing attributes LOWER_LEFT and UPPER_RIGHT in the symbol at the corresponding locations.

`Can't find a symbol file for` *symbol_name*`.`

The symbol name was referenced but could not be found. This error usually occurs because the search path for symbols is wrong.

`Can't open command` *filename*`.`

A –c option for *filename* could not access the file.

`Can't open` *filename*`.`

A command option with *filename* could not access the file.

`Can't open output file` *filename*`.`

The output file could not be opened, probably because the output directory is missing or protected, or the disk space is full.

`Can't open WIR input file` *filename*`.`

The netlist input file cannot be found or is protected.

`Cannot find sheet` *symbol_name*`.`

The symbol specified in a –Sheetsym command option cannot be found in the symbol search path. Either the name has been entered correctly, or the symbol search path is wrong.

`Cannot read megafile` *symbol_name*`.`

An error occurred while reading a symbol from a megafile. The megafile is probably corrupted.

Component `has no connections.`

A component with no connected nets was found. This error may have occurred because a global net was deleted or because an input connector has an unused net.

Component `is not defined in WIR file` *filename*`.`

An attribute references an undefined component. The component definitions should precede the attribute definitions for the component.

`Defining implicit I/O pins from non $ nets.`

This is a warning message indicating that the input netlist file does not contain input or output pin symbols, and a symbol for the schematic cannot be found. As a last recourse, all named nets, that is, nets other than the default internal named nets beginning with $, are assumed to be external and have an input or output connector added.

`Error reading symbol file` *filename*`.`

An error occurred while reading a symbol file, probably because of a corrupted file or megafile.

Filename `is not a WIR netlist file.`

The header in the input file does not match the format for a WIR file.

`File path concatenation is too long.`

The file name and path name must be less than 256 characters long. Either reduce the directory nesting, or rename the files and directories with shorter names.

`I/O connector` *symbol_name* `does not have 1 pin.`

The pin symbols that are assumed to be input/output connectors must have only one pin.

`Improper` *-option_name* `option.`
`Improper command option` *-option_name*`.`

A problem occurred with *–option_name,* typically because of a missing parameter or a non-numeric parameter.

`Improper permutability definition for` *symbol_name*`.`
`Improper permute definition for` *symbol_name*`.`

The permutability string in a –Permute option for *symbol_name* has an error.

`Improper sheet name` *name*`.`

A –Sheet, –Minsheet, or –Maxsheet option has an invalid sheet name.

`Input connector` *symbol_name* `does not have an output pin.`

An input connector should have a single pin defined as an output on the right side of the symbol.

`Input file` *filename* `too long.`

The file name and path name must be less than 256 characters long. Either reduce the directory nesting, or rename the files and directories with shorter names.

Instance_name `already defined in WIR file line number` *line_number*`.`

An instance definition appears more than once within the netlist input file, probably because the input file is corrupted.

`Insufficient pins on instance name symbol name, WIR file line number` *line_number*`.`

The component specified references a symbol defined as having more pins than specified on the component, probably because the symbol search path is wrong and the wrong version of a symbol is referenced, or the symbol definition has been modified.

`Mismatched nesting in permute definition for` *symbol_name*`.`

The permutability string in a –Permute option for *symbol_name* has an error. The nesting of ( ) or [ ] does not match or is unbalanced.

`Net` *net_name* `is not defined in WIR file line number` *line_number*`.`

An attribute references an undefined net. The net definitions should precede the attribute definitions for the net.

`Net for I/O pin` *pin_name* `cannot be found.`

The symbol for the schematic contains the pin name that represents a net not in the input WIR file.

`No components are defined.`

The netlist input file does not define any components.

`No I/O connections are defined.`

The netlist input file does not define external pins.

`No nets are defined.`

The netlist input file does not define any nets.

`No symbol files can be found.`

No symbol files were found in the symbol search path, probably because the search path is incorrect.

`No symbols are defined.`

The netlist input file does not reference any symbols.

`Out of memory attempting to allocate` *number* `bytes.`
`Out of memory attempting to reallocate` *number* `bytes.`

All available memory was used up attempting to create the schematic.

`Output connector` *symbol_name* `does not have an input pin.`

An output connector should have a single pin defined as an input on the left side of the symbol.

`Permute definition for` *symbol_name* `has improper pin name.`

The –Permute definition for *symbol_name* has an error.

Pin_name `on` *symbol_name* `does not have a proper input/output type.`

Pin_name in *symbol_name* does not has a proper I/O type, probably due to a missing PINTYPE attribute on the pin.

Pin_name `can't be found for` *symbol_name* `in WIR file line` *line_number*.

The symbol referenced in an AS record specifies a undefined pin in the symbol. This error has probably occurred because the symbol search path is wrong and the wrong version of a symbol is referenced, or the symbol definition has been modified.

`Sheet size overflow: suggest -levellimit` *value* `-heightlimit` *value*.

The –Levellimit or –Heightlimit option used to control the page splitting did not yield a result that fit within the sheet. This error can occur from the default values for some designs and could occur for improper user-specified values. The values shown in the message suggest different parameter values. If the –Restart option is not zero, the program is automatically restarted.

Symbol_name `cannot be found.`

Symbol_name was referenced but cannot be found. This error usually occurs because the search path for symbols is wrong.

Symbol_name `has no pins.`

Symbol_name is not an annotation symbol and does not have pin definitions. This error could have occurred because the symbol search path is wrong, and an improper version of the symbol was read.

Symbol_name `is not defined for instance name, line number.`

Symbol_name was referenced but could not be found. This error usually occurs due to an incorrect search path for symbols.

Symbol_name `needs` *number* `pins in WIR file line` *line_number*`.`

The specified component references a symbol defined as having a different number of pins than specified on the component. This error could have occurred because the symbol search path is wrong and the wrong version of a symbol is referenced, or the symbol definition has been modified

`Syntax error in symbol file` *filename* `line` *line_number*`.`

The symbol file read for *symbol_name* has an error in line *line_number*, probably because the file or megafile is corrupted.

`Syntax error in WIR input line` *line_number*`.`

The netlist input file being read has an error in the specified line number, probably because the file is corrupted.

`System error.`

This message indicates a system error. You should not receive this type of an error message under normal circumstances.

`System problem splitting a page; could not find any components.`

ViewGen could not find any components to fit on a page. This error might be caused by an extremely small –Heightlimit value.

`Too much nesting in permute definition for` *symbol_name*`.`

The permutability string in a –Permute option for the symbol name has an error, which may be due to an internal system error.

`Unexpected end-of-file for symbol file` *filename*`.`
`Unexpected end-of-file for WIR file` *filename*`.`

The file read for symbol file *filename* or WIR file *filename* name has an end-of-file error, probably because the file or megafile is corrupted.

WIR2XNF

WIR2XNF may issue the following warning and error messages.

Warning Messages

WIR2XNF may issue the following warning messages.

`Warning 16. The XNF attribute` *attribute* `on` *net_instance* `or` *pin_instance symbol_type* `is either not legal for the PARTYPE family or the XNF version. It will be treated as a USER attribute.`

`Warning 17. Duplicate parameter values` *attribute* `and` *attribute* `on EXT net` *net_name*`.` *Value* `will be used.`

There were duplicate attributes found on the indicated net. Attributes were placed on both the I/O pad and the net connecting the I/O pad and I/O buffer. Locate the net on the schematic, and edit or modify the attributes by selecting the net and then selecting **`Change`** → **`Attr`** → **`Dialog`** → **`All`** from the menu.

`Warning 27. Attribute` *attribute_name* `on` *symbol_name* `has no value. Ignored.`

The specified attribute has no value associated with it. WIR2XNF ignores it.

`Warning 31. Different pintypes [`*pin_type1*` and` *pin_type2*`] on SYMBOL and COMPONENT of pin` *pin_name* `of` *component_name*`. Using SYMBOL value.`

An attribute that was specified on a component has values in both the symbol and component values. The program automatically uses the value specified in the symbol value. If that is not the attribute that you want to use, you must modify the symbol and edit the attribute.

`Warning 32. Duplicate` *attribute_name* `attributes on equivalent nets` *net_name1* `and` *net_name2*`.`

The indicated nets have the same attribute. Because the nets are equivalent, having duplicate attributes is not necessary. To avoid this warning message, locate one of the nets on the schematic and edit the attribute. For hierarchically equivalent nets, only the top-most net attribute is required.

`Warning 33. PIN attribute` *attribute_name* `being ignored on` *symbol_name* `because not 2000 part type.`

The indicated attribute on the symbol is used only for the Xilinx XC2000 family part. For more information on the valid attributes for each family, see the "Attributes" section in the "Design and Simulation Techniques" chapter of this manual.

`Warning 69. Missing part type for -p option. Option ignored.`

When the –p option is specified, the next parameter must be the part type, for example, 4005PG156-5. Re-enter the command using an appropriate part type. See *The Programmable Logic Data Book* for a listing of legal part types.

`Warning 70. Unknown option -`*option* `ignored.`

The option specified on the command line prompt is not valid. See the section "WIR2XNF Options" in the "Program Options" appendix of this manual, or type **`wir2xnf`** to see a list of the valid options and their functions.

`Warning 71. Extra argument` *argument_name* `ignored.`

An extra argument was used on the command line when WIR2XNF was invoked. See the section "WIR2XNF Options" in the "Program Options" appendix of this manual for more information, or type **`wir2xnf`** at the command line to see a list of the available options and their functions.

`Warning 72. Command line PART=`*parttype1* `covers schematic PART=`*parttype2*`.`

The part type specified on the command line and the part type specified on the schematic are different. The program uses the part type specified on the command line. If you would like to use the part type specified on the schematic, do not use the –p option on the command line when invoking WIR2XNF.

Error Messages

WIR2XNF may issue the following error messages.

`Error 1. File extension` *extension* `does not have any valid file name characters.`

The input or output file name extension on the command line is invalid. Re-enter the command without specifying extensions, or specify the .1 extension for the input file and the XNF extension for the output file; these are the defaults.

`Error 3. Expected file name to have extension` *extension*`.`

The input file does not have the appropriate extension for the file to be processed. Check that the specified file name is correct. Re-enter the command without specifying file name extensions, or use the suggested extension for the file name.

`Error 4. Internal initialization failure.`

The internal initialization failure is a failure in the Viewlogic ViewBase subroutine. Consult Viewlogic for more information.

`Error 5. Could not open file` *filename* `for` *function*`.`

The file name indicated could not be opened for the indicated function. Be sure that you have a write privilege for this file and that it is located in the appropriate directory.

`Error 7. Illegal PIN to PIN connection (`*pin_type1* `to` *pin_type2*`) on NET` *net_name*`.`

The indicated net is connected to components that have conflicting PINTYPE attributes. This message usually indicates that an illegal connection has been made. Locate the net on the schematic and correct the violating connection. For more information on valid pin type connections and PINTYPE attributes, see the "Attributes" section in the "Design and Simulation Techniques" chapter of this manual.

`Error 9. Illegal NET to BLOCK connection (`*pin_type1*` to `*pin_type2*`) on NET `*net_name*`.`

The indicated net is connected to a hierarchical symbol whose pin type conflicts with the lower-level component pin type. This message usually indicates that an illegal connection has been made. Locate the net on the schematic and correct the violating connection. For more information on valid pin type connections and PINTYPE attributes, see the "Attributes" section in the "Design and Simulation Techniques" chapter of this manual.

`Error 10. Net `*net_name*` has sense conflict in hierarchy.`

The indicated net is active-High and active-Low at different levels of the hierarchy. Change the schematic so that the sense of the net is consistent throughout the hierarchy.

`Error 11. Could not resolve parametric attribute `*attribute_name*` on instance `*instance_name*`.`

The parametric attribute, for example, @PULSELO, specified on the indicated instance is not resolvable. A value must be assigned to parametric attributes to translate correctly. Edit the attribute by selecting the instance in PROcapture and executing the **`Change`** → **`Attr`** → **`Dialog`** → **`All`** command to bring up a dialog box. Check to be sure that the value assigned is valid.

`Error 22. Invalid PINTYPE attribute value `*value*` on pin `*pin_name*` of component `*component*`.`

The value of the PINTYPE attribute on the specified pin on the specified component is not one of the valid pin types for the FPGA environment. Edit the schematic or the symbol of the component to locate and change the value of the PINTYPE attribute to an appropriate value. The acceptable values are given in the

"Attributes" section of the "Design and Simulation Techniques" chapter.

`Error 28. Invalid BUS attribute value` *value* `on` *component*`.`

This is an internal error caused by corrupt libraries. Reload the libraries.

```
Error 30. No part type. Part type MUST be
specified in design or on command line.
```

Viewlogic Interface Guide

Altran

Viewlogic Interface Guide — 0401307 01

Printed in U.S.A.

Appendix D

Altran

You may need to port a design from one architectural family to another when using the Unified Libraries. To do so, you must change the library name in your viewdraw.ini file and the library alias with which each component on the schematic and in the library is prefixed. Viewlogic offers the Library Alias Maintenance Facility to assign these aliases and to change them. This utility is called Altran in Powerview and XAltran in PRO Series. You can use Altran or XAltran to change the aliases for both libraries and designs that already use library aliases, or you can use it to assign library aliases to proprietary libraries that you use in conjunction with Powerview or PRO Series.

This appendix describes how to run Altran manually in Powerview. Instructions for running XAltran in PRO Series are given in the "Design Entry" chapter.

Library Aliases

A library alias is a short text string that identifies and references a specific library or project directory. Every Viewlogic-supplied library is assigned a library alias. Each time that you place a component in a design, ViewDraw prefixes the component's name with its library alias. ViewDraw determines each library's alias from its reference in the viewdraw.ini file.

In Viewdraw.ini File

To use Altran to change component aliases in the schematic from XC3000 to XC4000, you must add a line in the viewdraw.ini file to reference the XC4000 library. Each DIR string in a viewdraw.ini file identifies a directory available to the project using that specific viewdraw.ini file. The library alias text string is enclosed in parentheses and appears to the right of each library directory path in the viewdraw.ini file.

Following is an excerpt from a viewdraw.ini file that is configured to convert an XC3000 design to an XC4000 design.

```
DIR [r] /powerview_path/unified/xc3000  (xc3000)
DIR [r] /powerview_path/unified/xc4000  (xc4000)
DIR [r] /powerview_path/unified/builtin (builtin)
```

The DIR keyword identifies the various libraries available to the project using this viewdraw.ini file.

In WIR File

Each component in a ViewDraw design is written to a WIR file. Each component in the WIR file is identified by its component name, but various library vendors may use the same component name for similar components. In addition, each vendor may assign different functional and electrical characteristics to similarly named components. A NAND2 gate in one vendor's library may function differently from a NAND2 gate in another vendor's library. To identify each component in the design explicitly, ViewDraw places each component's library alias in the design's WIR file. The format for component names using library aliases is the following:

library_alias **:** *component_name*

For example, a library named nec24 is assigned "nec" as its library alias. If you place an AND2 gate from the nec24 library into a design, its name appears as "nec:and2" in the design's WIR file.

The following is a portion of a WIR file for the 74ls00 part from the Powerview 74ls library. The 74ls00 is built from components contained in the Powerview builtin library. The WIR file shows the builtin alias prefixed to the component name.

```
V4.1
K201946930700741s00
DW test741s:741s00
AS builtin:nand2 PINORDER=QN A B
AP builtin:nand2 1 PINTYPE=OUT
AP builtin:nand2 2 PINTYPE=IN
AP builtin:nand2 3 PINTYPE=IN
M builtin:nand2 $118
```

Library Privileges

You can use Altran to update or change the aliases for any library or design, but you must have read and write privileges for those libraries to do so. You must have read and write privileges from the viewdraw.ini file, as well as from the operating system.

You can add parameters to the DIR keyword in your viewdraw.ini file to specify the libraries for which you have read and write privileges. Depending on the format of your viewdraw.ini file, there are two ways to determine the read and write privileges for a particular library.

- If you use the older numbered DIR format, numerals 1 through 5 are designated by Powerview as user libraries, indicating that you have read and write privileges to those directories. The following is an example of a DIR line from a viewdraw.ini file that uses the numbered format to identify library privileges:

  ```
  DIR 3 /libs/nec (nec5)
  ```

  If you use this format, simply create a new temporary viewdraw.ini file that specifies the previously unprivileged directories as DIR 1 through 5.

- If you use the new Powerview DIR format, all libraries designated with [w] give you read and write privileges to that directory. The following is an example of a DIR line from a viewdraw.ini file that uses the newer lettered format to identify library privileges:

  ```
  DIR [w] /libs/ts24ac (tos24)
  ```

  If you use this new format, simply change the privileges to [w].

Updating or Changing Aliases

To update or change an alias using Altran, type in the following syntax from the operating system command line:

altran *option* [*existing_alias=new_alias*]

Option can be one of the following options:

- –all
- –l
- –n
- –p
- –s
- –nocheck
- –v

These options are described in the "Options" section later in this appendix. You normally use the –s option to convert your schematics. You can enter only one of the first five options each time that you run Altran, but you can use –nocheck and/or –v with any of the five. Use *existing_alias=new_alias* only when you want to change an existing alias name. The "Options" section provides examples of the syntax just given.

Altran updates the viewdraw.ini file each time that it changes or updates aliases. It does not delete the original viewdraw.ini file but renames it to viewdraw.bak. All the alias information produced by Altran is written to the viewdraw.ini file, but the original settings are saved in the Altran-created viewdraw.bak file.

Removing Aliases

You cannot selectively remove aliases from a project, library, or schematic page. Altran only allows you to remove all the aliases.

Altran uses two wildcards to form a single statement to remove aliases from the specific library, project, or schematic page. This statement is the following:

```
ALL%=NULL%
```

The ALL% wildcard searches for all the aliases in the specified libraries, projects, or schematic pages. The NULL% value is equivalent to "no alias."

If you have aliases in a library, and your viewdraw.ini file appears as the following:

```
DIR [W] /74ls (vl74ls)
```

and you enter the following command syntax:

```
altran -l vl74ls ALL%=NULL% -nocheck -v
```

Altran deletes *all* the aliases from all the components for each schematic in the vl74ls library. For example, schematics that referenced builtin:nand2 would only reference nand2. The viewdraw.ini file appears as the following:

```
|The alias vl74ls was translated to no alias
DIR [w] /74ls
| DIR [w] /74ls (vl74ls)
```

Options

The following options are available in Altran.

–p

The –p option updates or changes the aliases for the components in a specific project. It hierarchically updates or exchanges all components in all the schematic sheets in the project. It reads the aliases from the viewdraw.ini file and prefixes the components in the project with the proper alias. Use the project name as the argument to the –p option. You can optionally specify a different alias other than that specified in the viewdraw.ini file by entering that alias as another argument to the –p option. Following is an example using the –p option:

```
altran -p cleo nc404
```

In this example, Altran places the "cleo" alias on all the components in the nc404 project.

–all

The –All option updates or changes the aliases for all the read/write libraries in your viewdraw.ini file. Read and write libraries are identified by the [w] parameter of the DIR keyword in Powerview format; they are libraries 1 through 5 in formats prior to 6.0. This option only updates or changes the components in libraries; it does not update or change designs.

–l

The –l option updates or changes the aliases for the components in a specific library. It does not update or exchange hierarchically; it only operates on the top-level schematics in a library. To update a library, use the library's alias as the argument to the –l option. Here is an example:

```
altran -l lsi_10k
```

Following is an example showing how to change the library alias for the components in a library:

```
altran -l lsi_10k builtin=buzz
```

Like the –All option, the –l option also only updates or changes libraries. It does not update or change designs.

–n

The –n option is similar to the –l option. It updates or changes the aliases for the components in a specific library but does so by referencing the library's DIR number. Like the –l option, it does not update or exchange hierarchically; it only operates on the top-level schematics in a library. To update or change a library with the –n option, use the library's DIR number as it appears in the viewdraw.ini file as the argument to the –n option. Here is an example:

```
altran -n 5
```

In this example, Altran updates the library designated as DIR 5 in the viewdraw.ini file using the library alias in viewdraw.ini for that library.

–s

The –s option updates or changes the aliases for the components on a specific schematic. It reads the aliases from the viewdraw.ini file and prefixes the components in the schematic sheet with the alias. Use the sheet name as the argument to the –s option. You can optionally specify a different alias other than that specified in the viewdraw.ini file by entering that alias as an another argument to the –s option.

Following is an example of the –s option:

```
altran -s c24.1
```

In this example, Altran places the aliases on the components on sheet c24.1 using the library aliases from the viewdraw.ini file.

–nocheck

After updating or changing the aliases, Altran by default runs the Check program to ensure the connectivity of your WIR files. The –Nocheck option tells Altran not to run the Check program.

–v

The –v option instructs Altran to display all the read, write, and error messages as it processes the files.

Using XC3000 Drawings for XC4000 Designs

If you want to use an existing XC3000 design drawn in ViewDraw as the basis for a new XC4000 design, take the following into consideration.

There are five symbols in the XC3000 library that cannot be converted to an XC4000 device. These are CLB, CLBMAP, IOB, OSC, and GXTL. Since each of these symbols depends on the architecture of the XC3000 family, they must be removed from the schematic and replaced with logic symbols. The XC4000 family does not have the on-chip crystal oscillator represented by OSC and GXTL, but the internal configuration oscillator can be accessed during operation by using the OSC4 symbol.

Viewlogic Interface Guide

Index

Index

XILINX®

The Programmable Logic Company℠

0401307

Printed in U.S.A.