VeriWell™ User's Guide 3.0

Preliminary

Wellspring Solutions, Inc.
7 Tudor Dr, Suite 300
Salem, NH. 03079
(603) 898-1100
Fax: (603) 893-8058
info@wellspring.com

http://www.wellspring.com

Notice

Software License Agreement and Warranty

Wellspring Solutions, Inc.’s complete Software License Agreement, its conditions, terms, warranty and limitations, as they relate to the purchase and use of Wellspring Solutions, Inc.’s software products, is found on the following page of this document. Customers are expected to read this agreement before breaking the seal on the package containing the software.

References to Features and Products

This document may contain references to products, functions, and capabilities not yet announced or available. If you have questions about the suitability of this product for specific applications and the availability of features and functions, please contact Wellspring Solutions, Inc.

References to Other Vendors’ Products

This document may refer to products and features of vendors other than Wellspring Solutions, Inc. Where such references exist, every effort has been made to ensure the validity of the reference at the time of publication. Wellspring Solutions, Inc. makes no warranty of any kind as to the performance or features of another vendor’s products.

Shareware

This document may reference software known as shareware. Shareware is a concept where software is obtained on a trial basis. Payment is made to the author after it is determined that the software is useful and will continue to be used. Where such references exist, you are encouraged to register the shareware with the creator if you decide to use it.

Wellspring Solutions, Inc., Software License Agreement

The Agreement constitutes the complete agreement between you, the licensee, and Wellspring Solutions, Inc. Carefully read all the terms and conditions, disclaimer and limited warranty of this agreement prior to opening the sealed media packet contained in this package. Opening the sealed media packet indicates your acceptance of this Agreement. If you do not agree with these terms and conditions, return the sealed media packet and any other materials that are a part of this product within 10 days to Wellspring Solutions, Inc. and your money will be refunded. No refunds will be given for products that have an opened media packet or missing components.

License

The enclosed software product which includes media and supporting documentation (the **Software**), is owned by Wellspring Solutions, Inc. and is protected by copyright law. The Software is provided to you by Wellspring Solutions, Inc. grants you a personal nontransferable and non-exclusive license to use the Software as set forth below:

You may:

- freely copy, distribute, or transmit the Software provided you do so only as a complete package, with all files included, and provided you keep intact and display all notices that refer to this License and the copyright notice.

You may not:

- rent, lease, or resell copies of the Software or documentation to others;
- modify or adapt the Software in whole or in part including but not limited to translating or creating derivative works;
- disassemble or reverse compile for the purpose of reverse engineering the Software; and
- tamper with the hardware key (if included) with the Software.

Term

The Agreement and your license to use the Software will automatically terminate without notice if you fail to comply with any provision of the Agreement. Upon termination, you shall destroy all copies of the Software. All disclaimers of the warranties and limitations of liability set forth in the Agreement shall survive any termination of this Agreement.

Selection and Use

You assume full responsibility for the selection of the Software to achieve your intended results and fix the installation, use and results obtained from the Software.

Limited Warranty

For a period of 90 days from the date of the receipt of the Software, Wellspring Solutions, Inc. warrants that:

- the media on which the Software is distributed and the documentation are free from defects in materials and workmanship; and
- The software will substantially conform to Wellspring Solutions, Inc.’s published specifications and to the documentation provided with it when used as specified in such documentation.

However, Wellspring Solutions, Inc. does not warrant that the functions contained in the Software will meet your requirements or that the operation of the software will be uninterrupted or error free. In the case of defect in material or non-conformance, and provided you return the item with a copy of your receipt within the 90-day period, Wellspring Solutions, Inc. shall at its discretion either: (a) correct the non-conformance of the Software; (b) replace the defective Software and/or documentation; or (c) refund the license fee.

Limitation of Remedies and Liability

If failure of any disk has resulted from accident, abuse, or misapplication, Wellspring Solutions, Inc. shall have no responsibility to replace the disk, or refund the purchase price. Any replacement disk will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This warranty gives you specific legal rights. You may have other rights, which vary from state to state. Except as expressly provided previously, the Wellspring Solutions, Inc. Software Product and printed documentation are provided **As Is**. Wellspring Solutions, Inc. does not make any warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event will Wellspring Solutions, Inc. be liable to you for any direct, indirect, consequential, or incidental damages (including damages from loss of business profits, business interruption, loss of business information, and the like) arising out of the use of or inability to use the Wellspring Solutions, Inc. Software Product even if Wellspring Solutions, Inc. has been advised of the possibility of such damages or for any claim by any other party. In no case shall any direct or indirect suppliers of Wellspring Solutions Inc. bear an liability for any reason whatsoever and in no case shall Wellspring Solutions, Inc.'s liability exceed the amount of the license fee actually paid by you.

General

This Agreement is governed by the laws of the Commonwealth of Massachusetts, except for copyright matters which are covered by United States laws and international treaties. Use, duplication, or disclosure by the U.S. Government of this computer software and documentation shall be subject to the restricted rights under DFARS 52.227-7013 applicable to commercial computer software. Export (Domestic Versions): Regardless of any disclosure made by Licensee to Wellspring Solutions, Inc. of an ultimate destination of the Software, Licensee shall not re-export or transfer the Wellspring Solutions, Inc. Software to anyone outside the United States without first obtaining a license from the U.S. Department of Commerce, as required.

If you have any questions about the Agreement, direct them in writing to: Wellspring Solutions, Inc., 7 Tudor Dr. Suite 300, Salem, NH 03079, USA.

CONTENTS

Part I: Introduction

Section 1: Getting Acquainted

This chapter provides VeriWell users with important and useful information about Wellspring Solutions, Inc., their products, and services.

Quick Start

This section will help you get started so you do not have to read the entire manual right away.

Step 1: Create a Project

VeriWell requires that all models be contained in a "project". This is true even if he model consists of one file. If you already have some Verilog files that you want to simulate, create a project and add the file or files that comprise your model. If you are starting from scratch, create a project and then open a new text file which will put you into VeriWell's syntax-coloring editor.

To create a new project, select the **project|new** menu item. Select the name and location of the project file. A blank window will pop up. Eventually, this will list the pathnames of the individual files that comprise the project.

Now, select the **project|add** file menu. This asks you to find the Verilog text files. By convention, the names of these files end with the ".v" suffix.

When you are finished selecting all of your Verilog files, you may close the project. Closing the project is the only time that the project is saved.

To reopen a recently-used project, select the file menu. The last four projects are listed and can be selected.

Step 2: Simulate

To run a simulation, press the arrow button in the console window. To stop, press the stop button.

Other buttons:

Start over

Continue simulation after a stop

Exit simuation

Step

Step and trace

Step 3: Using Waves

To view waves, you will need to put the following line in an initial block somewhere in your model:

```
$vw_dumpvars;
```

When your simulation reached this statement, VeriWell will open the waves window. At first, the window will be blank. You will need to select which signals to view and in which groups.

To select groups and signals, press the browser button ▤. This will put you into the browser window. Before you can select signals, you will need to make a new group. Press the **new group** button and type in whatever you want to call the group.

Now use the left side to add signals. If you are using VeriWell/Free, you will only see 10 signals. See detailed description of $vw_dumpvars to show how you can select particular signals to be captured.

Use the right and left mouse buttons to select regions in the waves to measure and zoom. Note that the cursors snap to the nearest edge.

Step 4: Save the Waves

Select the **edit|preferences|waves** menu item. Check the **Group File Enabled** box. This will remember the groups every time you reenter the project.

Now for the rest of it…

About the package

VeriWell is distributed electronically and by floppy disk. The manual is also in electronic form, distributed in Portable Document Form (PDF). A hardcopy is available from Wellspring Solutions. Contact Wellspring for pricing and availability.

Readme File

Users are recommended to read the **Readme** file found on the software media included with the product package to reference the latest information that may not have been available at the time the User's Guide was published.

VeriWell/Free and Copy Protection

Most versions of VeriWell are copy-protected in some way. VeriWell for Windows, for example, requires that a hardware key be plugged into the printer port of the PC. VeriWell for Sparc requires a special code be provided on the command line. Unlike most copy protected programs, however, the absence of the protection mechanism does not cause VeriWell to stop working.

If the copy protection mechanism is missing VeriWell will run, however the size of the source module is limited. This mode is referred to as VeriWell/Free.

VeriWell/Free is VeriWell without the copy protection. There is one executable program (for each operating environment). With the copy protection enabled, it runs unlimited-sized models and the Preference modules owned by the user. Without the copy protection, VeriWell runs approximately 500 lines of source code, not including comments and whitespace, and with most Preference functionality enabled. The VeriWell executable (VeriWell/Free) is freely distributable. It is available the **Wellspring web pages**. (Instructions for downloading are provided later in this section.) The free version can be used for small designs, to learn Verilog, or to evaluate VeriWell. It is not shareware—there is never any obligation to upgrade to the registered version.

For registered users, the version of VeriWell that is posted at the Wellspring Solutions, Inc.’s Internet Home Page represents the latest version of the software and can be downloaded periodically and used as an upgrade.

Registering VeriWell

Customers should register their software by filling out the pre-addressed and pre-stamped registration card included in the product package and mailing it to Wellspring Solutions, Inc. Because many copies of the software are sold to end users through agents and distributors, Wellspring Solutions, Inc. will not have you in its records until you register. Once you are registered, we can keep you informed of the following:

- Latest product development news;
- Availability of new product releases; and
- Other announcements.

Customer Support

If you purchased your product directly from Wellspring Solutions, Inc., and you need technical assistance, you can reach someone almost immediately by email at support@wellspring.com.

Or, you can call a leave a message at our main number, (603) 898-1100.

If you purchased your Wellspring Solutions, Inc. product from a distributor or reseller, please contact the distributor or reseller who sold you your product.

Bug Reports

Wellspring Solutions, Inc. welcomes comments and reports on bugs that VeriWell users may encounter. To fix the problem, the technical staff needs to reproduce it. Therefore, if you are reporting a bug, please send the whole module or code fragment of the model via E-mail to support@wellspring.com.

Although we cannot actively support users of VeriWell/Free, we encourage them to contact us if they find a problem with VeriWell.

Ordering Wellspring Solutions, Inc.'s Products

Wellspring Solutions, Inc.'s products can be ordered directly in the United States by calling our toll-free telephone number:

800-VERIWELL (**800-837-4935**), 9 A.M. **-** 5 P.M. eastern time

or send email to sales@wellspring.com.

Additionally, Wellspring Solutions, Inc. has several distributor sites located outside the United States. See our web site for details.

Receiving Upgrades

Wellspring Solutions, Inc. regularly improves current revisions of the software. These upgrades are free and can be obtained from the Wellspring Solutions, Inc.'s web site.

World Wide Web Site

The Wellspring Solutions, Inc.'s homepage contains information on upgrades, releases, beta tests and free downloads of VeriWell for Windows 95/NT, Sparc and other environments. The homepage is located at the URL: http://www.wellspring.com.

Electronic Mail (E-mail)

Wellspring Solutions, Inc. will send upgrades to customers by E-mail, however, this method is used only on a need basis because of the large files that need to be sent. Some internet providers will not allow large files to be sent. Some email utilities may not understand how to handle binary files.

Regular Mail

Upgrades can also be sent on distribution media by regular mail. If you are not able to gain access to any of the electronic methods described previously (FTP, Web Site, or E-mail) or these methods are not convenient for you, call 1-508-865-7271 to receive the upgrades by regular mail.

Recommended Publications

For a complete reference to Verilog HDL language, we recommend that you refer to the following publications.

- The Verilog **HDL** Language Reference Manual (LRM), IEEE-1364

Wellspring Solutions, Inc. highly recommends that VeriWell users obtain a copy of the LRM and use it in conjunction with the VeriWell User's Guide 2.0. The LRM is available from the Institute of Electrical and Electronics Engineers, Inc. (IEEE) via their web site, WWW.IEEE.ORG or by calling 1-800-678-IEEE or 1-908-981-9667.

- The Verilog Hardware Description Language, Third Edition

Written by Donald E. Thomas of Carnegie Mellon University and Philip R. Moorby, the creator of the Verilog language, this tutorial bundles a 500-line version of VeriWell/Free for MS-DOS and is published by Kluwer Academic Publishers, Norwell, Massachusetts, 1991;
ISBN 0-7923-9723-1.

Consider visiting one of the large online bookstores ans search for the word "Verilog".

Section 2: The VeriWell User's Guide

Chapter Two describes the scope, intent, organization, and content of this VeriWell User's Guide 2.0.

How to Use This Manual

The VeriWell User's Guide 2.0 is divided into two parts. Part I includes information about VeriWell, Feature List, and Installation guide for each operating environment. Part II provides general information on how to use and run VeriWell on each environment available. The Appendices contain The Verilog HDL Language Reference Manual, IEEE-1364 Cross Reference and Implementation Differences from Verilog-XL.

Intended Audience

This guide assumes that the reader already has a working knowledge of the Verilog HDL. It is not intended as a Verilog reference manual or a Verilog tutorial. If you are not familiar with Verilog, then you should obtain a copy of the Language Reference Manual (LRM), IEEE-1364. The LRM is a complete reference to Verilog HDL.

Scope of Manual

The purpose of this guide is to explain how to use the VeriWell simulator and its Preferences within the presently supported environments. It also describes the similarities and differences between the features of VeriWell, version 2.0, and Verilog HDL as defined in the Verilog Language Reference Manual (LRM), IEEE-1364.

Experienced XL users are advised to read the sections in this document detailing the differences between VeriWell and Verilog-XL. XL users can use the manuals from Cadence Design Systems, although there may be differences found in Cadence's manuals that are not documented in the VeriWell User's Guide.

VeriWell Release Notes

VeriWell Release Notes provide information on enhancements or changes to version 3.0 since the VeriWell User's Guide 3.0 was printed. These Release Notes are documented separately and may be included with the VeriWell product package. If your VeriWell product package does not contain a separate set of VeriWell Release Notes, then there have been no significant enhancements to be documented since your package was shipped.

If Wellspring Solutions, Inc. prepares Release Notes after you receive your package, we will automatically mail them to you so you can stay up to date. Additionally, these Release Notes can be found on FTP and Web Site along with the latest VeriWell version.

VeriWell Release Notes are intended to be either added to the User's Guide or replace specific pages or sections of the guide. Instructions on how to merge release note documentation into the User's Guide will accompany the set.

Section 3: About VeriWell and Feature List

Overview of Verilog HDL

Verilog HDL, a hardware description language used to design and document electronic systems, is the EDA industry's first standard HDL. It permits designers to describe their designs at higher levels of abstraction, such as architectural or behavioral, and provide a path to logic synthesis. Verilog HDL also permits for mixed-level designs, where users can describe their design at both high and low levels of logic simulation and synthesis.

Verilog HDL is the most widely used HDL with a user community of more than 35,000 active designers who are choosing top-down design and mixed-level design to contend with the ever-increasing design complexities and shrinking time to market cycles.

Verilog HDL was designed by Philip Moorby and introduced in 1985 by Gateway Design Automation. Moorby built a simulator around Verilog-XL in 1984-85, and continued to make a second major contribution at Gateway with the algorithm for every fast gate-level simulation. Gateway Design Automation grew rapidly with the success of Verilog-XL when it was acquired in 1989 by Cadence Design Systems, Inc.

A proprietary language until 1990, Cadence brought Verilog HDL to the public domain in May of 1991 with the formation of Open Verilog International (OVI), the organization that oversaw Verilog-related activities. Putting Verilog in the public domain opened the market for other Verilog HDL-related software companies to develop and expand resulting in broader acceptance of the language.

Verilog has been standardized by the IEEE. The Design Automation Sub-Committee was formed in 1993 to provide the IEEE Verilog standard 1364. Under this Sub-Committee is an active IEEE operating group which produced the Verilog standard IEEE-1364 in 1995. Wellspring Solutions Inc. has been an active participant in this process.

About VeriWell

VeriWell is a comprehensive, IEEE-compliant implementation of Verilog HDL. Through its development of VeriWell, Wellspring Solutions, Inc. has met Verilog users' needs for robust and affordable Verilog-HDL design tools. VeriWell includes integrated project management, an integrated editor, waveform viewer, and VERILOG Simulator.

VeriWell supports the Verilog language as specified by the Language Reference Manual, IEEE-1364. VeriWell was first introduced in December, 1992, and is the first independently-developed simulator to be written, from the first line of code, to be compatible with the IEEE-1364 standard and with Verilog-XL. Because it was developed on the PC, it was specifically designed to be memory-efficient with relatively high performance.

VeriWell is used by ASIC designers and consultants for all pre- and post-synthesis model development. At the behavioral and RTL levels, VeriWell is used to develop entire synthesizable models and test environments. At the structural level, VeriWell is used to verify netlists generated by either synthesis tools or schematic entry tools.

As a component of a large-scale top-down design methodology, VeriWell is used in conjunction with other high-end IEEE-compliant simulators, such as Verilog-XL or Chronologic's VCS, to develop behavioral, RTL and synthesis models in Verilog HDL. This complementary marriage between VeriWell and other simulators enhances the design process and greatly reduces the user's overall tool costs. Verilog-XL users will recognize the familiar user interface (single-step, trace, interactive commands) and the fact that VeriWell supports command files, input files, and log and key files.

In addition, the superior error-detection capabilities of VeriWell find most syntax and semantic errors on the first pass. This relieves the frustration of users who hitherto would fix all reported errors only to find a new set of errors reported.

Wellspring Solutions, Inc. is committed to providing comprehensive, compatible, and affordable design tools for Verilog users. To that end, Wellspring Solutions, Inc. will continue to upgrade and enhance the VeriWell simulator package to offer its customers high-quality products and timely technical support.

VeriWell Feature List

The following represents the definitive list of language constructs supported by VeriWell 3.0:

Data Types

Static types:

- REG
- REG arrays
- INTEGER
- INTEGER arrays
- TIME
- REAL
- MEMORIES
- PARAMETER

Nets:

- WIRE/TRI
- WOR/TRIOR
- WIAND/TRIAND
- TR10
- TR11

Operands

- number
- net
- register
- integer, time
- net bit-select
- register bit-select
- register part-select
- net part-select
- memory element
- function
- system function
- strings
- min:typ:max
- hierarchical names

Operators

- concatenation
- arithmetic +, -, *, /
- modulus
- relational > < >= <=
- logical negation
- logical and
- logical or
- logical equality
- logical inequality
- case equality
- case inequality
- bit-wise negation
- bit-wise and
- bit-wise inclusive or
- bit-wise exclusive or
- bit-wise equivalence
- reduction and
- reduction nand
- reduction or
- reduction nor
- reduction xor
- reduction xnor
- left shift
- right shift
- conditional (?:)

Built-in Primitives

- and
- nand
- nor
- or
- xor
- xnor
- buf
- not
- bufif0
- bufif1
- notif0
- notif1
- tran
- tranif0
- tranif1

User-Defined Primitives

- combinational
- sequential

Statements

- Continuous assignments
- Net assignments
- Procedural assignments
- Blocking procedural
- Non-blocking procedural assignments (<=)
- forever
- #delay
- repeat
- while
- for
- Intra-assignment delay
- defparam
- if-else
- case
- casex
- casez
- @
- @(posedge)
- @(negedge)
- begin/end
- fork/join
- named blocks
- always
- initial
- tasks
- functions
- disable
- assign/deassign
- force/release

Hierarchical Structures

- Port connections by ordered list
- Port collapsing
- Named Ports

- Hierarchical references

Specify Blocks

- $setup
- $hold
- $period
- $width
- $skew
- $recovery
- $setuphold
- $nochange
- specparam

System Tasks and Functions

- $display[bho]/$write[bho], • $showstats
  %h, %o, %d, %b, %c, %m, %x, • $wwaves
  %t, %s, %f, %e, %g • $random
- $fdisplay[bho]/$fwrite[bho] • $test$plusargs
- $strobe[bho]/$fstrobe[bho] • $dumpvars
- $monitor[bho]/$fmonitor[bho] • $dumpfile
- $monitoron/$monitoroff• $dumpon/$dumpoff
- $fopen/$fclose • $dumpflush
- $readmemh/$readmemb• $dumpall
- $time/$stime • $bitstoreal
- $stop, $finish [no arguments] • $realtobits
- $settrace, $cleartrace • $itor
- $scope, $showscopes • $rtoi
- $log/$nolog • $realtime
- $showvars • $timeformat
- $key/$nokey • $printtimescale
- $input

Command Line Options

- -c (compile only)
- -f (command argument)
- -i (input file)
- -s (stop)
- -t (trace)
- -l (log name). The log file can be disabled with the command line argument "-l nolog".
- -k (key name). The key file can be disabled with the command line argument "-k nokey".
- -v (library file)
- -y (library directory)

Compiler Directives

- `define
- `ifdef, `else, `endif
- `include
- `timescale

Predefined Plus Options

- +maxdelays/+mindelays/+typdelays
- +define+<macro name>+<macro name>...
- +synopsys (displays warnings for constructs unsupported or ignored by Synopsys HDL Compiler 2.x.)

- +noshow_var_change (disables tracking of location and time of each variable update)
- +libext+<string> (library file extension)
- +incdir+<directory>+... (include-file search rules)

Interactive Commands

- Interactive statements (compile and execute a normal behavioral statement)
- . (continue with the simulation)
- , (step and trace a single statement)
- ; (step a single statement)
- : (colon)

Limitation Summary

- Register and net vectors are limited to 262,080 bits
- Bit-qualified decimal numbers are limited to 32 bits (i.e. 32'd1234). Bit qualified numbers of other radii are limited to 262,080 bits.
- All expressions representing controls are limited to 32 bits. These are: delays, repeat counts, shift counts, array indices, and bit- and part-select indices.

Strengths, pullup/pulldown, trireg are not supported
Port collapsing is not supported
Vector net expansion is not supported.

Note: The indices for a vector may be any number up to 4G (the highest number represented in 32 bits), but the range must not be larger than 262,080. Delay expressions are 32 bits wide.

Section 4: VeriWell Installation

Chapter 2: VeriWell for Sparc

Introduction

This chapter describes the system prerequisites you must meet and the procedures you must follow to install and use VeriWell successfully on your Sparc.

System Requirements for Sparc

- Sparc and Sparc compatible systems using SunOS 4.0.x or greater (including Solaris versions)
- 8 megabytes (MB) of RAM
- 2 MB of hard disk space

What is Included

The contents of the files included with VeriWell for Sparc are described in your distribution media. VeriWell may be distributed in several ways. If you received the software electronically (FTP, Web Site, E-mail), then, most likely, all of the files are compressed into a single tared and compressed file.

Part II: Using VeriWell

Section 1: Using VeriWell with IDE

Overview

VeriWell for Windows, X, and Macintosh use a graphical front-end to easily control the simulation and debugging of VeriWell models. This permits for integrated editing and modification of Verilog models and the support of **projects** which are collections of files comprising a particular model.

The interfaces for both Windows and Macintosh versions are nearly identical and are therefore described together in this section. Where there are differences between the interfaces, they are explicitly mentioned.

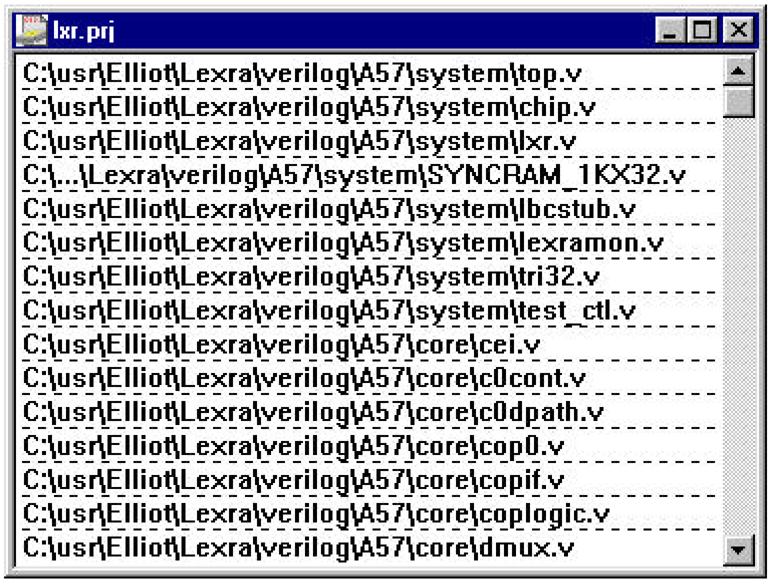
Working with Projects

For each module, there is a **project**. The project is the collection of source files and options that are required to run the simulation. A project keeps track of all the source files and stores information needed to debug, compile, and create the program. The following section explains how to create a new project, open or close an existing project, and add, rearrange, or remove files.

To simulate a module, first create a new project by selecting **New Project** from the **Project** menu.

The Project Window

The Project Window displays the text documents in your project. It lists the names of the text documents to be compiled and simulated.



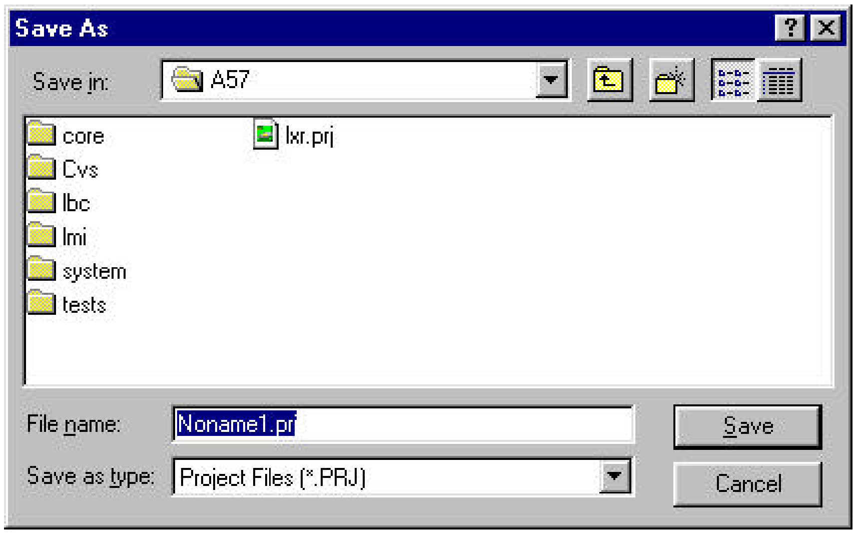
```
C:\usr\Elliot\Lexra\verilog\A57\system\top.v
C:\usr\Elliot\Lexra\verilog\A57\system\chip.v
C:\usr\Elliot\Lexra\verilog\A57\system\lxr.v
C:\...\Lexra\verilog\A57\system\SYNCRAM_1KX32.v
C:\usr\Elliot\Lexra\verilog\A57\system\lbcstub.v
C:\usr\Elliot\Lexra\verilog\A57\system\lexramon.v
C:\usr\Elliot\Lexra\verilog\A57\system\tri32.v
C:\usr\Elliot\Lexra\verilog\A57\system\test_ctl.v
C:\usr\Elliot\Lexra\verilog\A57\core\cei.v
C:\usr\Elliot\Lexra\verilog\A57\core\c0cont.v
C:\usr\Elliot\Lexra\verilog\A57\core\c0dpath.v
C:\usr\Elliot\Lexra\verilog\A57\core\cop0.v
C:\usr\Elliot\Lexra\verilog\A57\core\copif.v
C:\usr\Elliot\Lexra\verilog\A57\core\coplogic.v
C:\usr\Elliot\Lexra\verilog\A57\core\dmux.v
```

Figure 1: Project Window

The Project Window (Macintosh) can be resized by the grow box found at the lower-right corner. The Project Window can also be enlarged by the zoom box found at the upper-right corner. The Project Window (Windows) can be resized by using the **Maximize** or **Minimize** buttons that are found in the upper-right corner.

Creating a New Project

To create a new project, select the **New Project...** command from the **Project** menu. You will see a standard dialog box (Figure 3). To create a project file, type in the name and click **Save**. You can also create a new folder for your project by clicking **New Folder**.



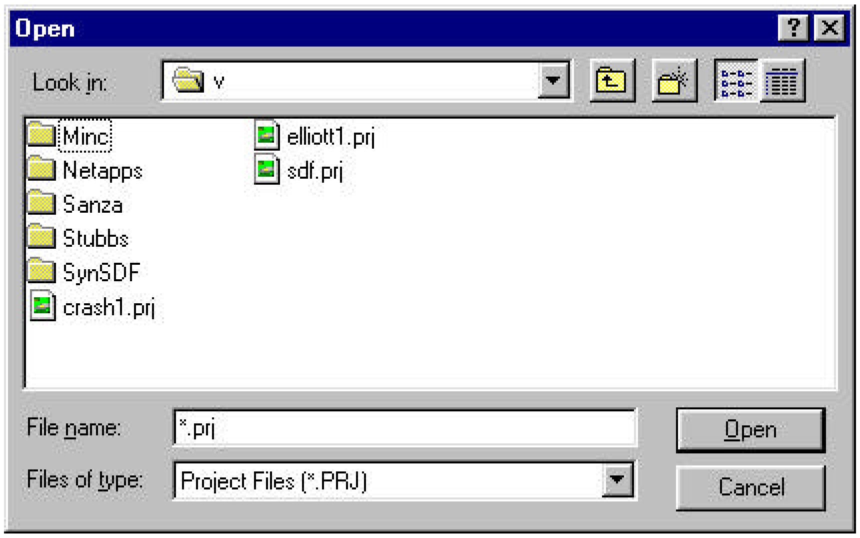
File name: Noname1.prj

Save as type: Project Files (*.PRJ)

Figure 2: Naming a New Project

Opening an Existing Project

To open an existing project, double-click its icon from the Finder, File Manager, or use the **Open Project ...** command from the **Project** menu.



File name: *.prj

Files of type: Project Files (*.PRJ)

Figure 3: Opening a Project

Under the **File** menu, you will also see a list of the last four projects opened.

Note: You can have only one project document open at a time.

Closing Projects

Use the **Close Project** command from the **Project** menu to close the project document. Any text documents that are open will remain open.

Adding Files to Projects

There are two methods for adding files to a project.

- If you are modifying a text document that is not included in the Project Window, you would use the **Add Window** command from the **Project** menu. This permits you to add the file in the current edit window to the project. If it is a new file, you must save the file with the **Save** command from the **File** menu before you can add it to the project.

- To add a file on disk to a project, select the **Add Files ...** command from the **Project** menu. You will see the dialog box (Figure 5). Only one file at a time may be selected to be added to the project. Choose the file you want to add and select **Open** from the **File** menu.

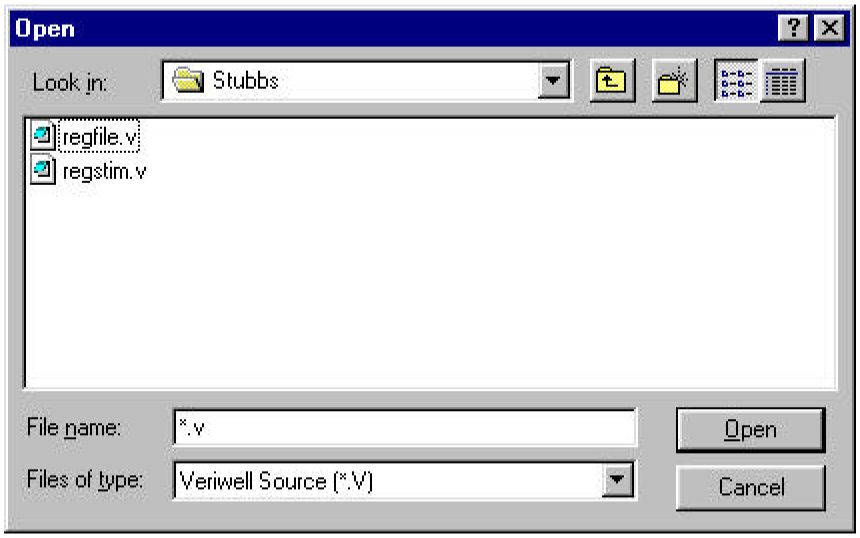
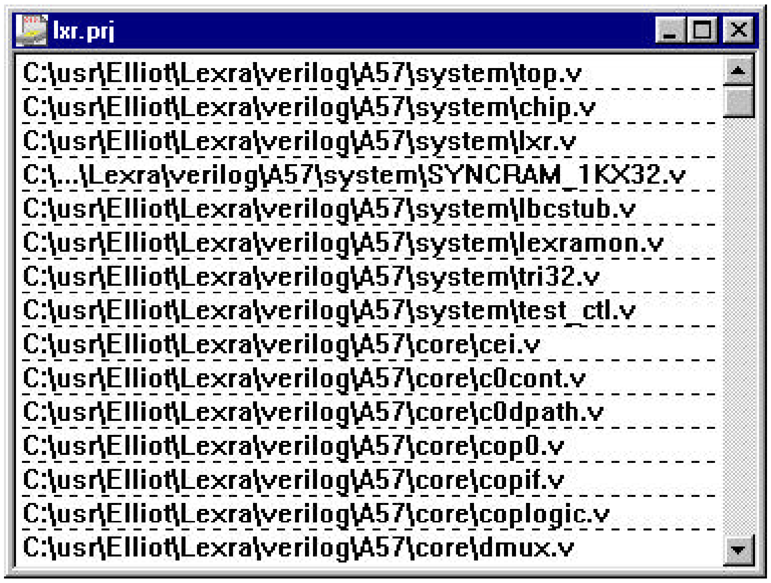


Figure 4: Adding Files to a Project

When a file is added to the project list, it is appended to the list. The files are compiled in the order listed. If the file name is too long for the Project Window, it will be shortened to fit. The file names are separated by a dotted line (Figure 6). To rearrange the list, select a file name and drag it to the position you want by using the mouse.



```
C:\usr\Elliot\Lexra\verilog\A57\system\top.v
C:\usr\Elliot\Lexra\verilog\A57\system\chip.v
C:\usr\Elliot\Lexra\verilog\A57\system\lxr.v
C:\...\Lexra\verilog\A57\system\SYNCRAM_1KX32.v
C:\usr\Elliot\Lexra\verilog\A57\system\lbcstub.v
C:\usr\Elliot\Lexra\verilog\A57\system\lexramon.v
C:\usr\Elliot\Lexra\verilog\A57\system\tri32.v
C:\usr\Elliot\Lexra\verilog\A57\system\test_ctl.v
C:\usr\Elliot\Lexra\verilog\A57\core\cei.v
C:\usr\Elliot\Lexra\verilog\A57\core\c0cont.v
C:\usr\Elliot\Lexra\verilog\A57\core\c0dpath.v
C:\usr\Elliot\Lexra\verilog\A57\core\cop0.v
C:\usr\Elliot\Lexra\verilog\A57\core\copif.v
C:\usr\Elliot\Lexra\verilog\A57\core\coplogic.v
C:\usr\Elliot\Lexra\verilog\A57\core\dmux.v
```

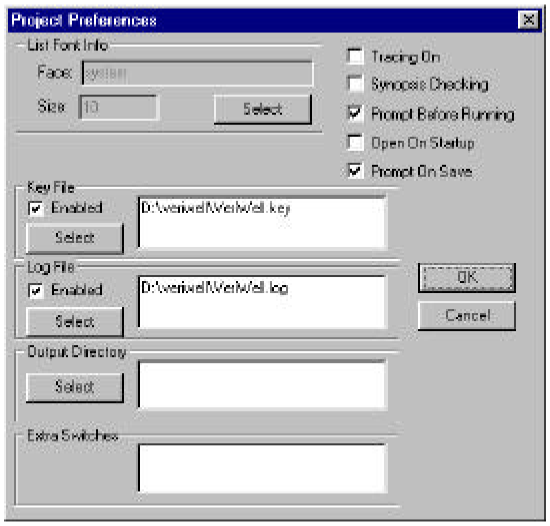
Figure 5: Project List

Removing Files from the Project

To remove a file from a project, select a file in the Project Window and select the **Remove File** command from the **Project** menu. This command only removes the file from the open project and will not delete the file from your disk. If a file is accidentally removed, it can be added back into the project.

Changing Project Preferences

To change any of the preferences for an opened project, select **Preferences**, **Project ...** from the **Edit** menu (Figure 7).


Project Preferences

List Font Info
Face: system
Size: 10

[ ] Tracing On
[ ] Synopsis Checking
[✓] Prompt Before Running
[ ] Open On Startup
[✓] Prompt On Save

Key File
[✓] Enabled
D:\veriwell\VeriWell.key

Log File
[✓] Enabled
D:\veriwell\VeriWell.log

Output Directory

Extra Switches

Figure 6: Project Preferences dialog box

- You can change the font type and size for the project list by using the **Font** and **Size** lists.

- To enable tracing, select the **Trace On** check box. When the simulation is run, every statement executed will produce a text output showing the file name and line number containing it.

- To enable a synopsis checking, select the **Synopsis Checking** check box.

- To pause for input before running a simulation, select the **Prompt Before Running** check box.

- To enable a key file, select the **Key File** check box. When running a simulation, every command entered will be written into the key file. The default key file is VeriWell.key. To specify a key file, type the path into the text box or use **Select** to select a key file by using a file selection dialog box.

- To enable the log file, select the **Log File** check box. When the simulation is running, output from the Console Window will be appended to the log file. The default log file is VeriWell.log. To specify a log file, type the path into the text box or use **Select** to select a log file using a selection dialog box.

- The **Output Directory** specifies the default folder where the VeriWell simulator will create output files. If the default folder is not specified, output files will be created in the folder containing the currently opened project file. To specify an output directory, use **Select** to select a folder. You can also type the path directly into the text box.

- **Extra Switches** are command line options to control simulation that can be entered in this text box.

- If you select the **Open on startup** check box, VeriWell will reopen any projects that were not closed when you last quit from VeriWell.

- When a project is run, if any project files have not been saved, a dialog box will prompt you to bring these files up to date. With the **Prompt On Save** check box selected, a dialog box will appear. With the **Prompt On Save** check box not selected, the project files will automatically be saved before running.

- **OK** will change the project preferences.
- **Cancel** will discard any changes made in the Project Preferences dialog box.
- **Use Defaults** will set all the project preferences to match the VeriWell application defaults.
- **Set Defaults** will set the VeriWell application defaults to match those currently displayed in the dialog box.

Creating a New File

To create a new file, select **New** from the **File** menu. An untitled edit window appears (Figure 8) with the cursor in the upper left corner ready for typing.

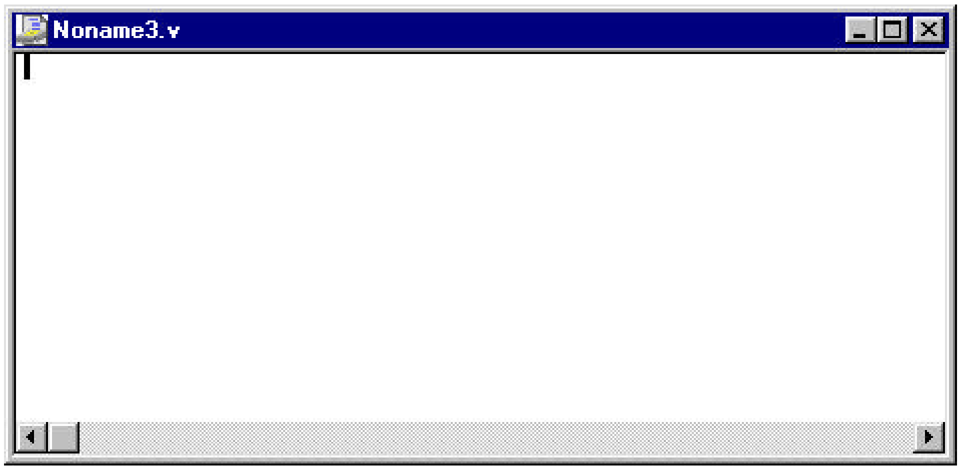


Figure 7: Untitled Edit Window

Opening a Text File

There are two methods for opening previously saved files.

- The **Open ...** command from the **File** menu opens any text document. A dialog box displays the names of all the text files in the current folder. The file selected will be shown in its own edit window. Under the **File** menu, you will also see a list of the last four files opened.
- A second way to open a text file that is already in your project is to double-click its name in the Project Window. If the file is already open, then double-clicking its name will bring the file's edit window to the front for modifying. From the **Windows** menu (Macintosh) or the **Window** menu (Windows) is a list of all opened text files and projects.

Editing a File

The editor for VeriWell employs all of the standard Macintosh and Windows editing techniques. Double-clicking on a word selects the entire word and triple-clicking anywhere in a line selects the entire line. **Cut**, **Copy**, **Paste**, and **Undo** all perform the same functions as in other Macintosh and Windows applications. The function keys **F1 - F4** and the Command Shortcuts will also work for **Cut**, **Copy**, **Paste**, and **Undo**. Refer to the Command Shortcuts for Macintosh and Windows found at the end of this chapter for a complete list of keyboard shortcuts.

Typing Text

The VeriWell editor does not have the wordwrap feature as in other editors. If you type past the right edge of the window, the window automatically scrolls horizontally so you will see the inserting point. You can use the horizontal scroll bar to see the beginning of the line.

Undoing Changes to a File

You can use the **Undo** command from the **Edit** menu to change the last thing you did. **Undo** will permit you to undo multiple operations with a limit that is set in the **Editor Preferences** dialog box. The wording of the **Undo** command will reflect the operation to be undone. If you have typed in a line, the command will read **Undo Typing**. If you do select **Undo**, the command will now say **Redo Typing**. There are also two shortcuts for **Undo**. You can use **COMMAND+Z** or the **F1** function key. You can use the command **Revert to Saved** from the **File** menu if you want to undo all the changes to a text file.

Selecting Words and Lines

To select a word, double-click on that word. If you double-click on a word and drag the mouse, you can extend the selection by a word. To extend an existing selection, hold down the Shift key and click to the end of the new selection. You can move an existing selection by clicking in the selection while holding the mouse down and dragging the selection. You can also duplicate a selection by clicking in the selection while holding down the **OPTION** key and dragging the selection. To select a line, triple-click on any point in that line. To select the whole file, choose **Select All** from the **Edit** menu or use **COMMAND+A**.

Indenting

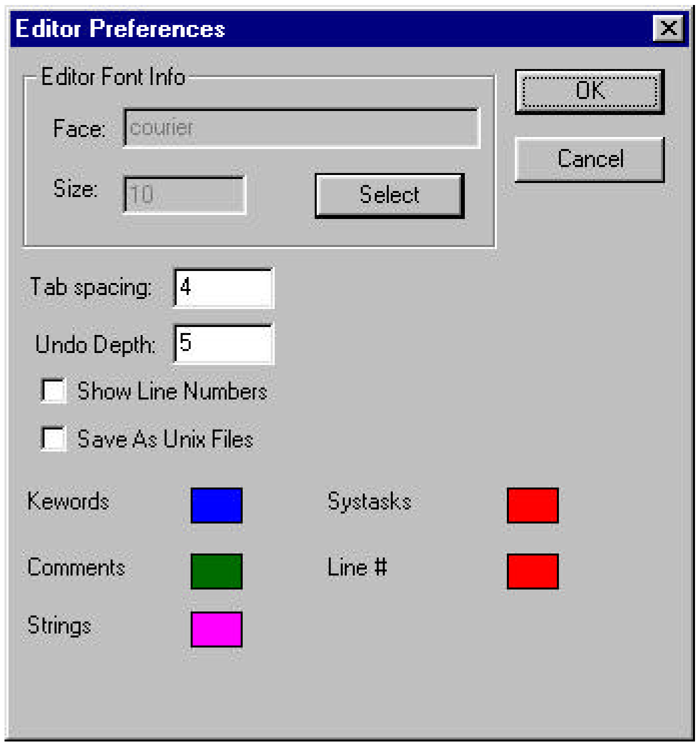
When you press **RETURN** to end a line, the editor indents the new line with the same number of leading tabs and spaces as the previous line. Backspacing changes the indentation. Tab spacing can be changed in the **Editor Preferences** dialog box.

Shifting Blocks Right and Left

You can change the indention level for a range of a line by using the **Shift Right** or **Shift Left** commands from the **Edit** menu. **Shift Right** inserts a tab at the beginning of each selected line while **Shift Left** removes the leading tab from each selected line.

Changing Editor Preferences

To change any of the settings, select **Preferences**, **Editor** from the **Edit** menu (Figure 9).



Face: courier
Size: 10
Tab spacing: 4
Undo Depth: 5
☐ Show Line Numbers
☐ Save As Unix Files
Kewords 🟦 Systasks 🟥
Comments 🟩 Line # 🟥
Strings 🟪

Figure 8: Editor Preferences dialog box

- The font type and size for the Editor Window can be changed by using the **Font** and **Size** lists.
- The **Tab Spacing** can be changed by typing in a new number in the Tab Spacing text box.
- **Undo depth** permits you to set the limit for how many operations can be undone. As in tab spacing, a new number can be typed in the Undo Depth text box.
- **Show Line Numbers** permits you to display line numbers in text files if this check box is selected.
- **Save as Unix Files** (Windows), if selected, permits you to save the text files with only a new line instead of a new line and return at the end of each line.
- **Keywords, Comments, Strings, Systasks, and Line #s** in text files can all be shown in distinct colors. If you want to highlight all keywords, you can double-click in the box beside keywords and a color wheel will appear. You can then select any color to use for highlighting.
- **OK** changes the editor preferences.

- **Cancel** discards any changes made in the Editor Preferences dialog box.
- **Use Defaults** sets all the editor preferences to match the VeriWell application defaults.
- **Set Default** sets the VeriWell application defaults to match those currently displayed in the dialog box.

Moving Around a File

The VeriWell editor permits several ways to move around the file quickly and efficiently through the use of Arrow and Function Keys. The Arrow Keys will move the cursor up, down, left, or right. At the end of a line, the right (or left) arrow key will go to the beginning of the next line (or end of the previous line). If you are using an Apple Extended Keyboard (Macintosh) or are using a standard PC keyboard (Windows), you can use the keys above the arrow keys summarized in the following table:

Press This	To Get This
HOME	Top of the file
PAGE UP	Back to previous screen
Forward DEL	Deletes the next character
END	End of the file
PAGE DOWN	Forward to next screen
SHIFT+RIGHT ARROW	Extends right selection by one character
SHIFT+LEFT ARROW	Extends left selection by one character
SHIFT+UP ARROW	Extends selection up by line
SHIFT+DOWN ARROW	Extends selection down by line
SHIFT+OPTION+RIGHT ARROW	Extends right selection by word
SHIFT+OPTION+LEFT ARROW	Extends left selection by word

The HOME, END, PAGE UP, AND PAGE DOWN will not move the insertion point, but merely scroll the file.

Moving to a Specific Line

The **Go To Line ...** or **COMMAND+MINUS SIGN** (-) from the **Search** menu permits you to move to a specific line in the file. Lines are numbered consecutively starting with the number 1.

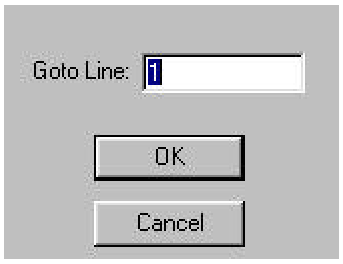


Figure 9: Go To Line dialog box

To go to a particular line, type in the line number and click **OK**. If you change your mind, you can click **Cancel**. The editor moves the insertion point to the beginning of the line you have chosen. While in the Console Window, you can triple-click an error or trace message to go to a particular line.

Saving a File

It is always a good idea to save your work periodically. To save a file without closing the file, you can use the **Save** command from the **File** menu or use **COMMAND+S**. If the file has not been saved before, you will be asked to name the file. Under the **Window** menu, there is a list of all opened text files. If changes have been made to a file since it was last saved, then a diamond will appear before the file name.

- The **Save As ...** command from the **File** menu permits you to save the current file under another name and then saves the contents of the edit window under that name. This feature is useful if you want to work with a new version of a file, while keeping the old file as a backup.

- To save all the open text files, use **Save All** from the **Windows** menu (Macintosh). This command works the same as saving each file individually.

Find a String

The VeriWell editor permits a variety of options for finding and replacing text within a file. Use the **Find ...** command from the **Search** menu or type **COMMAND+F** when you want to find a string. In the dialog box (Figure 11), you can specify a search string and an optional replacement string.

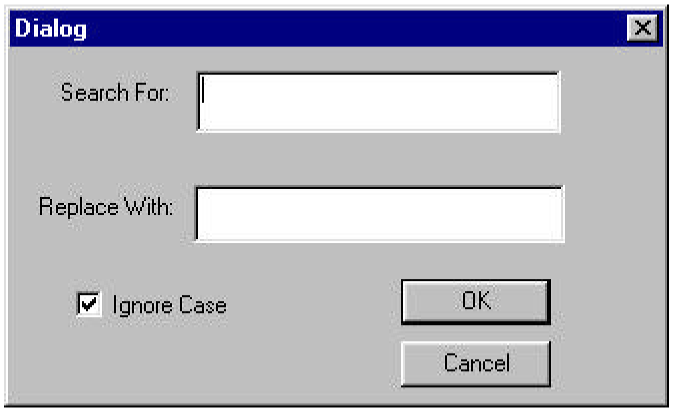


Figure 10: Search and Replace dialog box

You can type the find and replacement strings in the edit boxes and click **Find**. The editor will highlight the text if the string appears in the file. If it does not appear, the editor will beep. When the **Ignore Case** option is selected, the editor will match the search string regardless of case. To find the next instance of the string, use the **Find Again** command from the **Search** menu or use **COMMAND+G**. The search will only look in the active text file and not in all the text files of the project.

The **Enter Selection** command from the **Search** menu permits you to set the search string from the active window. If you select the search string in the active window, you can use the **Enter Selection** to place the string into the **Find ...** dialog box. You can then use **Find Again** to begin searching.

Replacing a String

If you would like to replace some but not all occurrences of the search string, enter a replacement string in the **Replace** box of the **Find ...** dialog box. When the editor highlights the first instance, you can use one of the following commands:

- **Find Again** or **COMMAND+G** to find the next occurrence of the search string.
- **Replace** or **COMMAND+EQUAL SIGN** (=) to replace the search string with a replacement string.
- **Replace & Find Again** or **COMMAND+H** to replace the current string and then immediately go to the next one.
- **Replace All** will replace all occurrences of the text for the entire text file. If you do not type in a replacement string, the editor will delete every occurrence of the search string.

Note: You cannot undo this command.

Printing Files

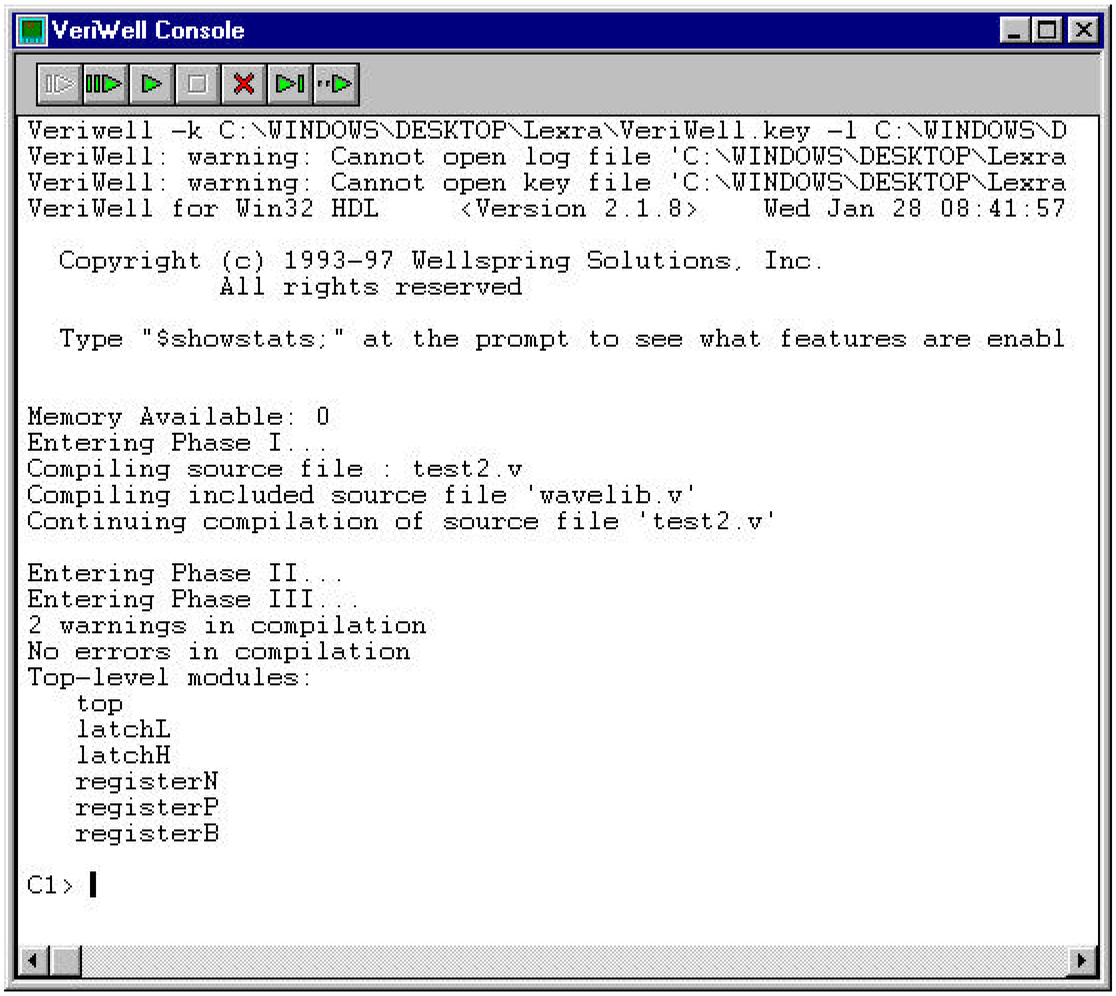
To print the contents of the front most window, select **Print ...** from the **File** menu and the standard printing dialog box will appear. You can print the contents of an editor window or console window. The **Page Setup ...** command from the **File** menu permits you to set the page size and other options before printing.

Closing a File

To close a file, click the window's close box (Macintosh), use the **Close** button (Windows), select the **Close** command from the **File** menu, or use **COMMAND+W**. If you have made any changes, the editor will ask if you want to save these changes before closing. The **Close All** command under the **Windows** menu (Macintosh) closes all open text files. If any file has not been saved, the editor will ask if you want to save it before closing.

VeriWell Console Window

When you start VeriWell, the Console Window (Figure 12) is opened immediately and is available for use. The window can be hidden by clicking in the close box of the window. To reopen the Console Window, select **Console** from the **Window** menu. The window will also be brought to the front if **Run** or **COMMAND+R** is selected from the Project Window. This method will only work if a project is open.



```
Veriwell -k C:\WINDOWS\DESKTOP\Lexra\VeriWell.key -l C:\WINDOWS\D
VeriWell: warning: Cannot open log file 'C:\WINDOWS\DESKTOP\Lexra
VeriWell: warning: Cannot open key file 'C:\WINDOWS\DESKTOP\Lexra
VeriWell for Win32 HDL     <Version 2.1.8>    Wed Jan 28 08:41:57

  Copyright (c) 1993-97 Wellspring Solutions, Inc.
            All rights reserved

  Type "$showstats;" at the prompt to see what features are enabl


Memory Available: 0
Entering Phase I...
Compiling source file : test2.v
Compiling included source file 'wavelib.v'
Continuing compilation of source file 'test2.v'

Entering Phase II...
Entering Phase III...
2 warnings in compilation
No errors in compilation
Top-level modules:
   top
   latchL
   latchH
   registerN
   registerP
   registerB

C1>
```

Figure 11: Console Window

Resizing

The Console Window (Macintosh) can be resized by using the zoom and grow boxes found in the right corners of the window. The Console Window (Windows) can be resized by using the **Maximize** or **Minimize** buttons that are found in the upper-right corner.

Entering Commands

There are three ways to enter commands:

- From the **Project** menu
- Using the buttons found at the top of the Console Window
- By typing commands into the Console Window

Entering Commands Using the Menu or Button Commands

The following is a brief description of the menu or button commands. The buttons will also be displayed 3D at times when these commands are legal.

Run	Start compilation or simulation
Restart	Restart compilation or simulation
Continu	Continue simulation
Stop	Stop simulation
Exit	Exit simulator
Step	Advances simulator one time step
Trace	Advances simulator one time step and traces lines executed

Entering Commands into the Console Window

To enter a command into the Console Window, type the command in the window and press **RETURN**. You can copy and paste text from the clipboard. The text will be pasted at the bottom of the window. By pressing **RETURN**, the text will be executed.

Console Output

The console will contain the output from the compilation and simulation of the project. By triple- clicking on an error message, the file containing the error will be opened with the error line highlighted. When tracing during a simulation, triple-clicking on a trace line will take you to the file and highlight the line being traced.

Finding a String

In the Console Window, VeriWell permits searching for text. Use the **Find ...** command from the **Search** menu or **COMMAND+ F** when you want to find a string. Type the string you are looking for in the **Search For** box and click **Find**. If the string appears in the Console Window, it will be highlighted. If it does not appear, the editor will beep.

If you want to find the next instance of the string, use the **Find Again** command from the **Search** menu or **COMMAND+G**. When the **Ignore Case** option is selected, the editor will match the search string regardless of case.

If you select the search string in the active window, you can use the **Enter Selection** to place the string into the **Find** dialog box. You can then use **Find Again** to begin searching.

Printing the Console Window

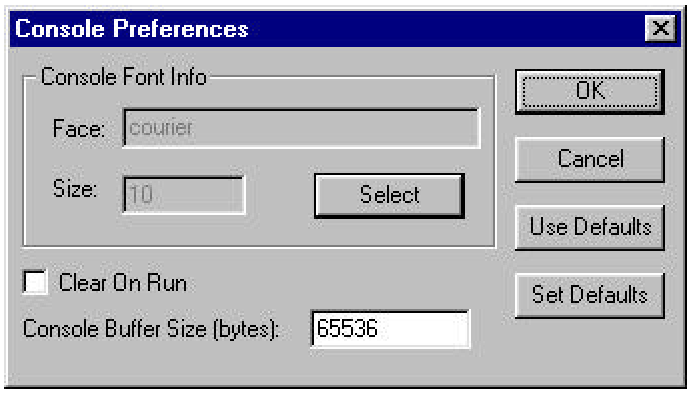
To print the entire contents of a console window, select **Print ...** from the **File** menu. The standard printing dialog box will appear. The **Page Setup...** command from the **File** menu permits you to set the page size and other options before printing.

Running a Simulation

When the **Run** command is chosen, the program will begin running the files in the project. If there are any files that are not up to date, a dialog box will appear. If you select **Yes**, the modified files will be saved before the run is started. If you select **No**, the files will not be saved, but the current displayed versions of the files will be compiled. If you select **Cancel**, no simulation will be run. You have this option for each modified file.

Changing Console Preferences

To change the preferences of the Console Window, you can select **Preferences**, **Console** from the **Edit** menu (Figure 13).


Console Preferences

Console Font Info

Face: courier

Size: 10

Clear On Run

Console Buffer Size (bytes): 65536

Figure 12: Console Preferences dialog box

- The font type and size for the Console Window can be changed by using the **Font** and **Size** lists.
- If you want the console clear before beginning the run, select the **Clear On Run** check box.
- The Console Window maintains a history of the output data. VeriWell permits you to scroll back to view this data, but this amount of data is limited. When it has reached its limit, characters at the beginning of the window are deleted. In this Preferences dialog box, you can set the **Console Buffer Size** (bytes).
- **OK** will change the console preferences.
- **Cancel** will discard any changes made in the Console Preferences dialog box.
- **Use Defaults** will set all the console preferences to match the VeriWell application defaults.
- **Set Defaults** will set the VeriWell application defaults to match those currently displayed in the dialog box.

Closing the Console Window

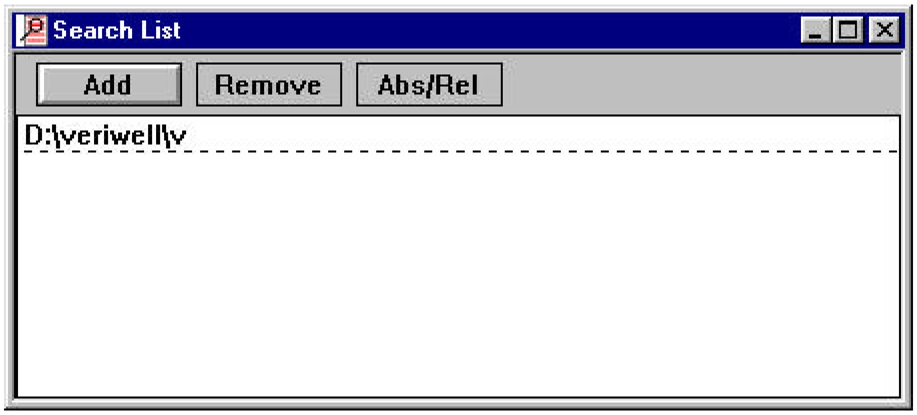
To close the Console Window, click the window's close box.

Search List Window

VeriWell permits each project to specify a set of folders to search for files opened by the VeriWell simulator. The Search List Window is used to add and delete directories from this list. The Search List Window also permits the search order to be controlled. This search list information is saved with each VeriWell project and will be restored whenever the project is opened.

Opening the Search List Window

The Search List Window is opened by selecting the **Search List ...** command from the **Edit** menu. This window is permited to be opened only when a project is opened. When the **Search List ...** command is executed, the Search List Window is opened (Figure 14).



D:\veriwell\v

Figure 13: Search List Window

The first entry of the Search List Window always contains an absolute path referencing the location of the currently opened project file. This entry cannot be deleted or modified. The folder is always the first location searched for any file to be opened by the VeriWell simulator.

Adding New Paths

Additional folders can be added by selecting **Add** in the upper left of the Search List Window. When this button is clicked, a dialog window (Figure 15) appears permitting a new folder to be selected that can be added to the list. To select a new folder, navigate to the folder to be added and then click **Search Here**. This new folder will always be added to the end of the list.

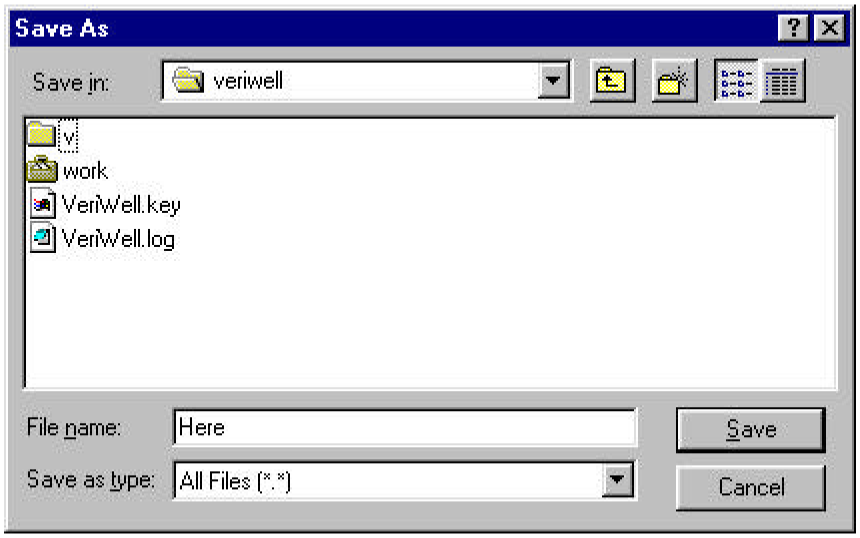


Figure 14: Adding Folders to the Search List Window

Folder Search Order

The order the VeriWell simulator uses to search folders is the same as the order the folders are listed in the Search List Window. The first folder listed is the first folder to be searched and the last folder listed is the last to be searched.

Changing the Search List Order

The Search List Window can be used to change the order that the folders are searched. An entry in the list may be dragged to the preferred location in the list by selecting the entry with the mouse, holding the button down and dragging the item to the preferred location in the list. When the button is released, this item will be moved to the selected location and the rest of the list will be moved down.

Absolute versus Relative References

The folders to be searched will be saved in the project file when it is closed. The folders' locations may be saved either as absolute references or as project relative references. Absolute references specify the disk the folder is on and the complete path to the folder to be searched. Project relative references are stored by referencing the folder, starting at the folder containing the project. Relative references must reside on the same disk as the currently opened project.

The advantage of relative references is that if the project and its associated folders to be searched are all contained in a common folder and that folder is moved to another machine or disk, the search paths all remain valid. Sometimes this is not preferred, such as a folder containing a library of files that is always in a fixed place, and in this case, absolute references should be used. When new paths are added and they are on the same disk volume as the currently opened project, they will always be added as project relative references.

The folders listed in the Search List Window may be changed between absolute and relative references by selecting the folder in the list with the mouse and then clicking **Abs/Rel**. If any item is double-clicked, it will be switched between absolute and relative references mode.

Removing Paths

Folders may be removed from the Search List Window by using the mouse to select the folder in the list and then click **Remove**.

Resizing the Search List Window

The Search List Window (Macintosh) can be resized by using the zoom and grow boxes found in the right corners of the window. The Search List Window (Windows) can be resized by using the **Maximize** or **Minimize** buttons that are found in the upper-right corner.

Closing the Search List Window

To close, click the window's close box (Macintosh) or use the **Close** button (Windows).

Waves Overview

Wellspring Waves is used concurrently with the I.D.E.-based versions of VeriWell. Waves is a full-featured, optional waveform viewing utility. It features smooth scroll, zoom, radix change, dynamic grouping, and full control of colors and fonts.

Using the Waveform Viewer

The Waveform Window is used to display a set of signals graphically. The Waveform Window may be opened in one of three ways.

- The **$vw_dumpvars()** system task is used during simulation to specify a set of signals to monitor. This system task is embedded within the Verilog source file. The execution of this system task will cause a waveform window to open and all signals will be monitored from the point where the system task is executed.

- The **Open Wave File** command from the **File** menu will open a previously saved Wave File.

- The **Convert Dumpvar ...** command from the **File** menu is used to convert a Verilog dumpvar file into a VeriWell Wave File and view it.

Displaying and Grouping Signals

When the Waveform Window is open, display groups may be set up by either loading an existing group file by using the **Load Group** command from the **File** menu, by embedding the **$vw_groupfile ("file name")** system task within the Verilog source file or by clicking **Browse**. The **Browse** button will open the Signal Browser Window which will permit the signal hierarchy to be traversed and a set of signals to be selected for viewing.

Customizing the Waveform Display

From the Waveform Window (Figure 16), you can control the simulation with the same buttons that are displayed on the Console Window. These buttons are run, restart, continue, stop, exit, step, and trace and are mentioned previously in this chapter under **VeriWell Console Window**.

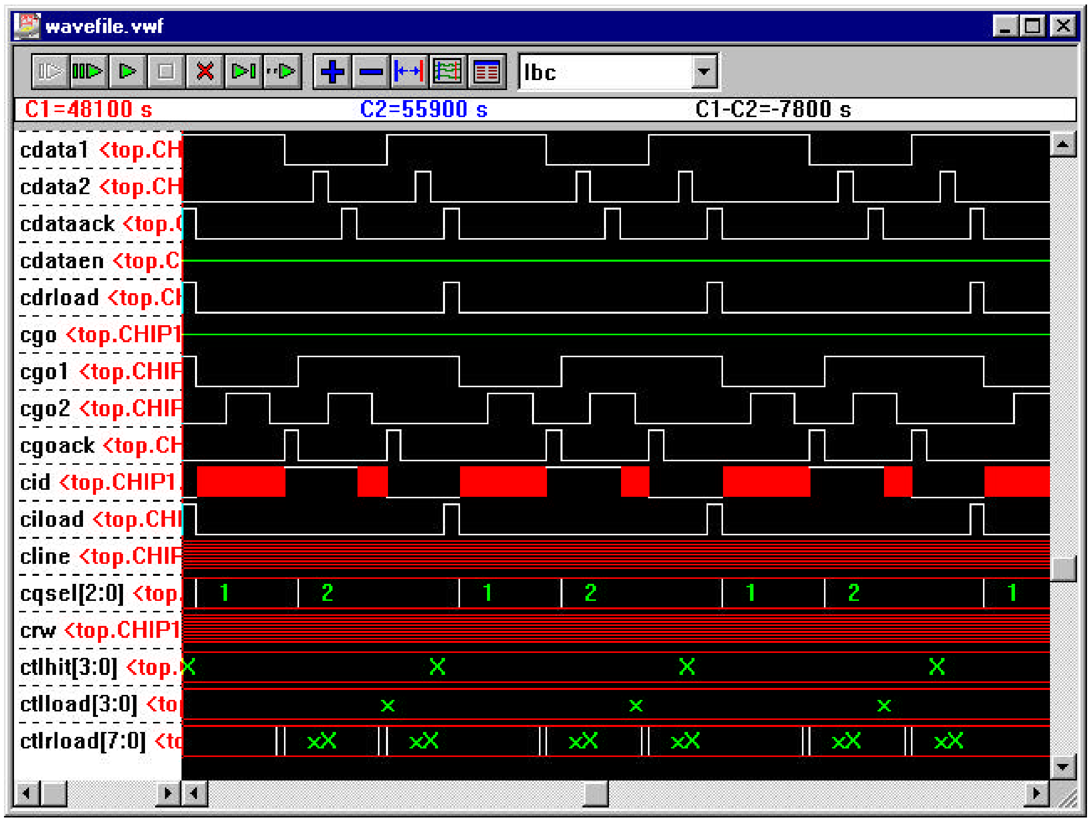


Figure 15: Waveform Window

Additional buttons:

- **Zoom In** will increase the display magnification 2x.
- **Zoom Out** will decrease the display magnification 2x.
- **Zoom To** will change the display magnification such that the region between the two cursors occupies the entire window.
- **Zoom Max** will open the **Signal Browser Window** permitting the creation, deletion, and modification of signal groups.
- **Combo Box Group List** selects the group of signals to be viewed. Signal groups may be created by using the signal browser, loading a group file, or during a simulation via the $vw_group ("file name") system task.

There are also two cursors in the waveform viewer. Cursor 1 moves by clicking the mouse or dragging and will snap to the closest signal event. Cursor 2 moves by holding down OPTION and clicking and will snap to the closest signal event. The colors of the two cursor can be changed by selecting **Preferences**, **Wave** from the **Edit** menu.

You can rearrange the order of the signals that are displayed by selecting a signal or group of signals and dragging them to a preferred location. Holding down SHIFT and clicking will select a range of signals and holding down OPTION and clicking will select a nonadjacent range of signals to move.

Double-clicking a selected signal name will display the value of the signal at the two cursors.

If you select a signal with width greater than 1, then you can set the radix of the selected signal. This is done by choosing **Set Radix** from the **Edit** menu.

The units for displaying time for the Waveform Window may be modified by choosing the **Set Time Unit** from the **Edit** menu. A dialog box will appear permitting the display units to be changed.

Changing Wave Preferences

The preferences for the waveform may be changed by selecting **Preferences**, **Wave** from the **Edit** menu (Figure 17).

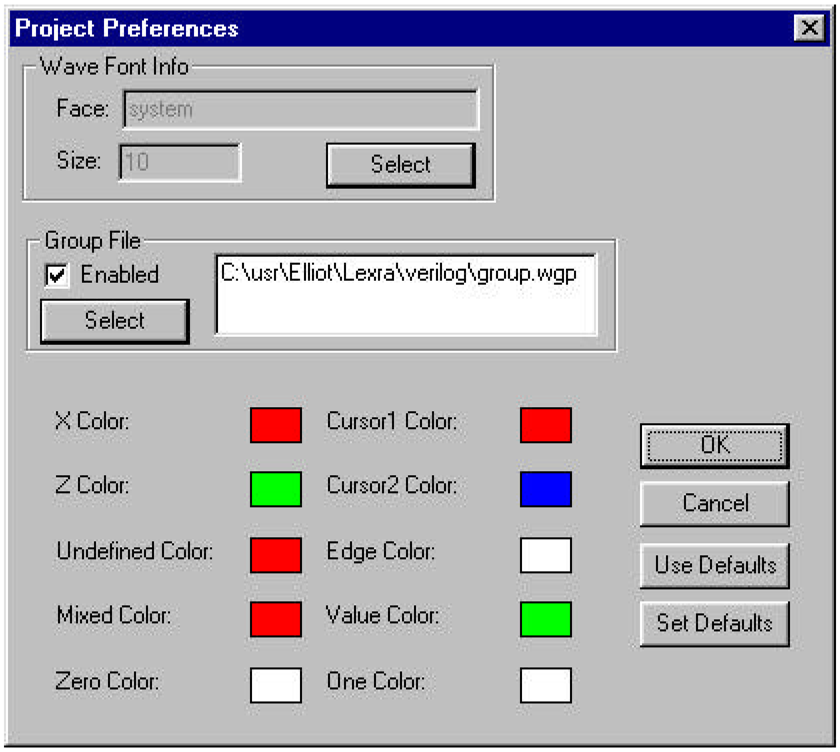


Figure 16: Wave Preferences dialog box

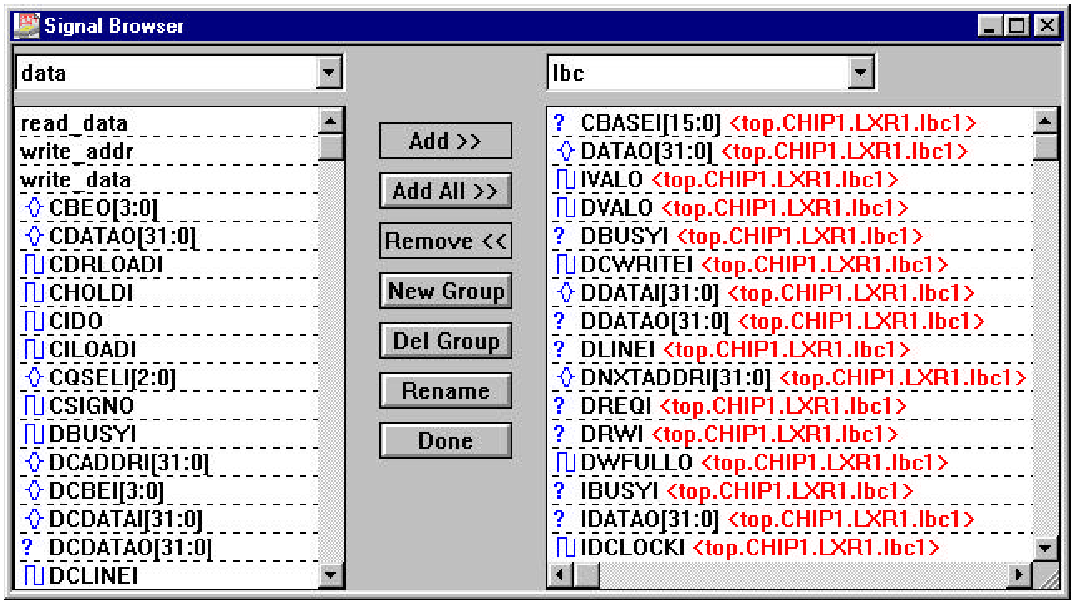
This window will permit the following options to be changed.

- The font type and size for the Waveform Window can be changed by using the **Font** and **Size** lists.
- **Group File** will open the group file that is named in the text box when the simulation is run. When you select the **Group File** box and click **Select**, a file selection window appears so that you can select a group file to be opened.
- **X, Z, One, Zero Color** changes the color of the displayed signals representing an X, Z, One, or Zero value.
- **Cursor1 and Cursor2 Colors** changes the cursor colors in the Waveform Window.
- **Undefined Color** changes the color for signals with undefined values.
- **Edge Color** changes the color of the vertical signal transitions.
- **Mixed and Value Color** changes the color for multi-bit signals.

- **OK** will change the wave preferences.
- **Cancel** will discard any changes made in the Wave Preferences dialog box.
- **Use Defaults** will set all the wave preferences to match the VeriWell application defaults. This button is only available if a project is open.
- **Set Defaults** will set the VeriWell application defaults to match those currently displayed in the dialog box. This button is only available if a project is open.

Signal Browser Buttons

When the Browser button is selected from the Waveform Window, the Signal Browser Window will open (Figure 18).



read_data
write_addr
write_data
CBEO[3:0]
CDATAO[31:0]
CDRLOADI
CHOLDI
CIDO
CILOADI
CQSELI[2:0]
CSIGNO
DBUSYI
DCADDRI[31:0]
DCBEI[3:0]
DCDATAI[31:0]
? DCDATAO[31:0]
DCLINEI

? CBASEI[15:0] <top.CHIP1.LXR1.lbc1>
DATAO[31:0] <top.CHIP1.LXR1.lbc1>
IVALO <top.CHIP1.LXR1.lbc1>
DVALO <top.CHIP1.LXR1.lbc1>
? DBUSYI <top.CHIP1.LXR1.lbc1>
DCWRITEI <top.CHIP1.LXR1.lbc1>
DDATAI[31:0] <top.CHIP1.LXR1.lbc1>
? DDATAO[31:0] <top.CHIP1.LXR1.lbc1>
? DLINEI <top.CHIP1.LXR1.lbc1>
DNXTADDRI[31:0] <top.CHIP1.LXR1.lbc1>
? DREQI <top.CHIP1.LXR1.lbc1>
? DRWI <top.CHIP1.LXR1.lbc1>
DWFULLO <top.CHIP1.LXR1.lbc1>
? IBUSYI <top.CHIP1.LXR1.lbc1>
? IDATAO[31:0] <top.CHIP1.LXR1.lbc1>
IDCLOCKI <top.CHIP1.LXR1.lbc1>

Figure 17: Signal Browser Window

From this window, signals can be added, removed, and groups created, deleted or renamed.

- The **Signal Display** list on the left side of the Signal Browser Window permits the model hierarchy to be navigated and the signals and modules of any given module to be displayed.
- The **Group Display** list on the right side of the Signal Browser Window permits the group lists to be navigated and signals in a given group to be displayed.
- **Add** will add the signal(s) selected in the signal display to the group currently selected in the group display.
- **Add All** will add all signals currently displayed in the signal display to the group currently selected in the group display.
- **Remove** will delete the currently selected signal(s) in the group display from the currently selected group display.
- **New Group** will display a dialog box permitting a new group to be created. The new group will become the currently selected group display.
- **Del Group** will delete the currently selected group display.

- **Rename** will display a dialog box permitting the currently selected group display to be renamed.

- **Done** will close the Signal Browser Window.

System Task Commands

The following system task commands can be embedded within the Verilog source file.

- **$vw_dumpfile** ("file name") is used to specify the waveform dump file name. If this system task is not used, the default file name is **wavefile.vwf**.

- **$vw_groupfile** ("file name") is used to restore a previously saved set of signal groups from **file name**.

- **$vw_group** ("groupname", signal1, signal2, ...) is used to create a new signal group named **groupname**. The list of signals will be added to the group. Only signals that have been probed via the $vw_dumpvars( ) system task may be added.

- **$vw_jumptotime** (time) is used to scroll the Waveform Window to a specific time.

- **$vw_dumpvars** (level, module1 or signal1, module2 or signal2, ...) is used to specify the set of signals to probe for later viewing. When $vw_dumpvars is specified without any arguments, all signals are probed. When $vw_dumpvars is specified with arguments, the first argument indicates how many levels below each specified module instance to probe. Either modules or signals may be specified. When the level is specified to be 0, all levels below the listed modules will be probed. Signals must be probed to be viewed. All signals will be saved in a file that can be opened after the simulation is complete or viewed as the simulation is run.

Veriwell Menus

File Menu

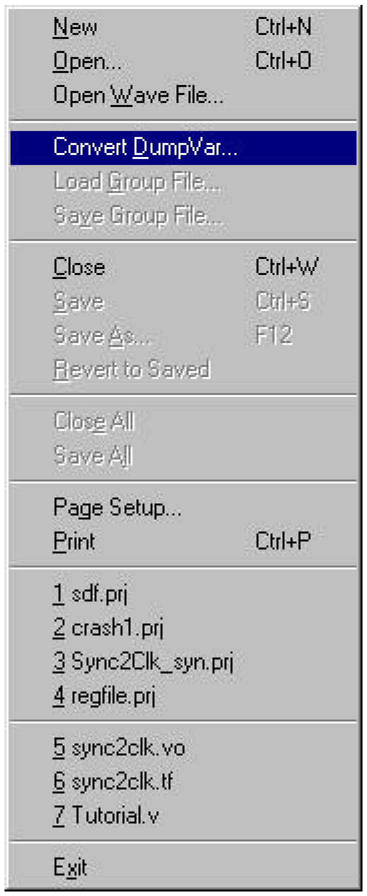


Figure 18: File Menu

New

This command will open a new untitled window. This file must be saved if you intend to add it to a project list.

Open...

This command will display a file selection dialog box permitting any text file to be opened and modified. A file that is in the currently opened project file list may also be opened by double-clicking on the name in the Project List Window.

Open Wave File ...

This command will display a file selection dialog box permitting an existing wave file to be opened and viewed.

Convert Dumpvar ...

This command will display a file selection dialog box permitting an existing dumpvar file to be opened and converted to a wave file.

Load Group File ...

This command will display a file selection dialog box permitting an existing group file to be loaded, initializing a set of signal display groups.

Save Group File ...

This command will display a file selection dialog box permitting the current signal display groups to be saved.

Close

This command will close the currently active window. It performs the same function as clicking in the window close box (Macintosh) or using the **Close** button (Windows). If you try to close a file and it has been modified, you will be prompted with a dialog box asking whether you would like to save the changes, discard them, or cancel the **Close** command.

Save

This command will save the active window to disk. If the window is currently untitled, you will be prompted to name the file.

Save As...

This command permits you to save the active window under a different name. The contents of the original file will remain unchanged. This is useful if you want to work with the new version of the file, while retaining the original file as a backup.

Revert to Saved

This command will discard any changes made to the active window, restoring it to the last saved version.

Close All

This command will close all open text files. If you try to close a file and it has been modified you will be prompted with a dialog box asking whether you would like to save the changes, discard them, or cancel the **Close** command.

Save All

This command will save all open text files.**Page Setup...**
This command will open the standard **Page Setup...** dialog box which permits the selection of paper size and orientation.

Print...

This command will open the standard **Print ...** dialog box permitting the specification of the number of pages and copies to be printed. Printing may be canceled by typing **COMMAND+PERIOD** (**.**).

Quit (Macintosh) **Exit** (Windows)

This command quits VeriWell and returns to the Finder or Windows shell.

Edit Menu

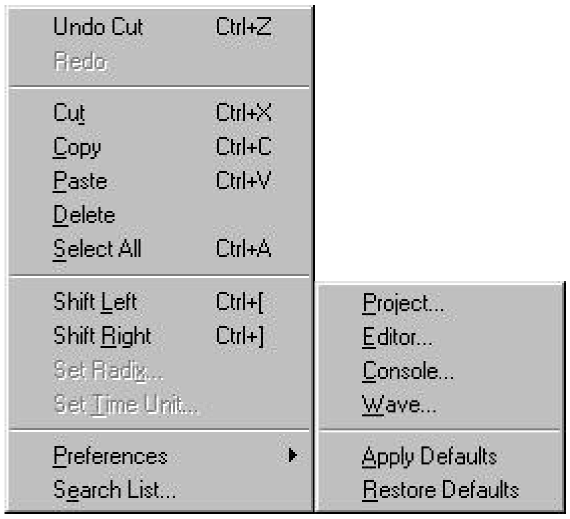


Figure 19: Edit Menu

Undo
This command will undo the last command. VeriWell supports multiple levels. The number of levels maintained is configurable in the **Editor Preferences** dialog box. The name of this command changes to reflect the operation that you are undoing. For example, after **Paste**, this command changes to **Undo Paste**. Once you have undone something, the name of the command changes to **Redo**.

Redo
This command will redo the last undone command. VeriWell supports multiple levels. The number of levels maintained is configurable in the **Editor Preferences** dialog box.

Cut
This command will delete the currently selected text, place it in the clipboard, and replace the current contents of the clipboard.

Copy
This command copies the selected text and places it in the clipboard. The copied text can now be pasted in your preferred location by using the **Paste** command.

Paste
This command will copy the contents of the clipboard and insert it into the active window at the current insertion point.

Clear
This command will delete the currently selected text.

Select All

This command will select all text in the edit window.

Shift Left

This command will shift the currently selected lines to the left by one tab space. Tab spacing can be changed in the **Editor Preferences** dialog box.

Shift Right

This command will delete the first character of the currently selected lines if they begin with a tab. Tab spacing can be changed in the **Editor Preferences** dialog box.

Set Radix ...

This command will display the **Set Radix** menu. This permits a signal's display radix to be set. This item is only enabled when a vector signal name is highlighted in the Waveform Window.

Set Time Unit ...

This command will display the **Set Time Unit** menu. This permits the units for displaying time to be set. The precision of simulation time is derived from the simulation. This item is only enabled when a waveform file is open.

Preferences

This command gives you control over the settings for the **Project**, **Editor**, **Console**, and **Wave** Windows.

Project...

This command will display the **Project Preferences** dialog box. This permits the project preferences such as font, tracing, synopsis checking, key and log files selections to be modified.

Editor...

This command will display the **Editor Preferences** dialog box. This permits the editor preferences such as font, tab spacing, line numbering, and highlighting to be modified.

Console...

This command will display the **Console Preferences** dialog box. This permits the console preferences such as font and console buffer size to be modified.

Wave ...

This command will display the **Wave Preferences** dialog box. This permits the wave preferences such as font, group file information, and wave display colors to be modified.

Apply Defaults

This command will set the current project preferences to match the VeriWell application default preferences.

Restore Defaults

This command will restore the VeriWell application default preferences to their factory settings.

Search List...

This command will open the Search List Window. This command is only available when a project is currently opened.

Search Menu

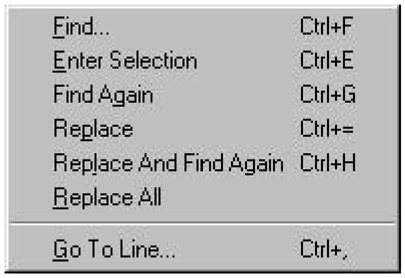


Figure 20: Search Menu

Find...

This command will open a dialog box permitting the entry of a find and a replacement string. Click **OK** to begin a search. If a match is found the text will be highlighted. When the **Ignore Case** option is selected, the editor's search and replacement functions disregard case when performing a search.

Enter Selection

This command will place the current text selection into the search string. You can use **Find Again** or **Find ...** to set search options.

Find Again

This command will find the next occurrence of the search string. If a match is found it will be highlighted. If it is not found, the editor beeps.

Replace

This command will replace the current selection with the contents of the replacement string.

Replace And Find Again

This command will replace the current selection with the contents of the replacement string. It then finds the next occurrence of the search string, but does not replace it. After each replacement, you will see the next instance of the search string, so you can decide whether to replace it. If you do, use the **Replace** or **Replace and Find Again** commands. If not, you can use the **Find Again** command to find the next occurrence of the string.

Replace All

This command will replace all occurrences of the search string with the replacement string.

Note: You cannot undo this command.

Go To Line...

This command will prompt for a line number in the open window, scroll to that line, and select it.

Project Menu

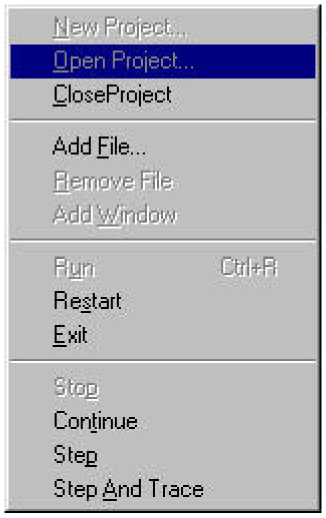


Figure 21: Project Menu

New Project...

This command will create a new project. You will be prompted for a name for this project. The **Project**, **Editor**, **Console** and **Wave** Preferences will be set to the VeriWell application default preferences.

Open Project...

This command will open a previously created project. You will be prompted to select a project file. You can only have one project document open at a time.

Close Project

This command will close the currently opened project. You will then be able to open an existing project or create a new project. When you close a project with open files that have been modified, but not saved, a dialog box prompts you to save them.

Add Files...

This command will display a file selection dialog box. Only one file at a time may be selected to be added to the project. The selected file will be added to the end of the opened project file list.

Remove File

This command will remove the currently selected file from the project list, but does not delete the file from your disk. If you remove a file accidentally, you can add it back into the project.

Add Window

This command will add the active window to the project file list.

Run

This command will compile and run the currently opened project. If any project files have not been saved, a dialog box will prompt you to bring these files up to date. If the **Prompt On Save** check box is not selected, then the project will be saved automatically before running. The **Prompt On Save** check box is found in the **Preferences, Project** dialog box.

Restart

This command will terminate the current selection, recompile, and run it. It is equivalent to interrupting the simulation, entering the **$finish;** command and issuing the **Run** command.

Exit
This command will terminate the current simulation. It is equivalent to interrupting the simulation and entering the **$finish;** command.

Stop
This command will interrupt the current simulation. It is equivalent to typing a **COMMAND+PERIOD** (**.**). After the simulation is interrupted, the user will be prompted for input.

Continue
This command will continue the simulation from the point it was interrupted. This command is equivalent to typing a period (**.**) character**.**

Step
This command will execute the next statement and return control to the user. This command is equivalent to typing a semicolon (;) character.

Trace (Macintosh)

Step and Step Trace (Windows)
This command will execute the next statement and print out a list of all simulation activity associated with the statement. Control will then be returned to the user. This command is equivalent to typing a comma (,) character.

Window Menu (Windows)

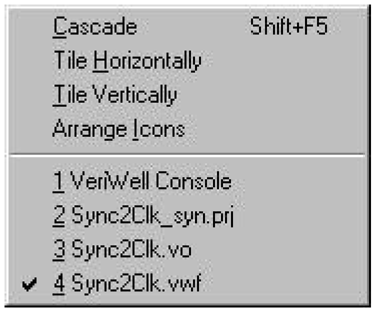


Figure 22: Window Menu

Cascade

This command arranges windows in an overlapping pattern.

Tile

This command arranges windows next to each other.

Arrange Icons

This command will rearrange the application icons.

Edit Windows

There will be one file name listed for each opened text window. The selection of a text window will bring it to the front.

Help (Windows)

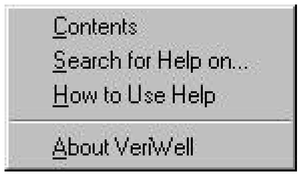


Figure 23: Help Menu

Contents

This command lists the entire contents of the Help file, the Introduction to **VeriWell Basics**, the **How To** list, and the **Commands and Buttons** list.

Search for Help on...

The **Search for Help on...** command permits you to type in a word you are searching for or select a word or phrase from a list. You can also select **Go To** topics.

How to Use Help

If you are new to Help, select **Help Basics**. You can also select a **How To** topic from an alphabetical list or click a topic from the **Commands and Buttons** list.

About VeriWell

This command will inform you which version of VeriWell you are using. When this command is selected, a dialog window is displayed with information about VeriWell and Wellspring Solutions, Inc. Clicking anywhere in the window will cause it to close.

Command Shortcuts for Macintosh and Windows

COMMAND+N	New
COMMAND+O	Open
COMMAND+W	Close
COMMAND+S	Save
COMMAND+P	Print
COMMAND+Q	Quit
COMMAND+X	Cut
COMMAND+C	Copy
COMMAND+V	Paste
COMMAND+A	Select All text
COMMAND+[	Shift Left selected lines one tab stop
COMMAND+]	Shift Right selected lines right one tab stop
COMMAND+F	Find
COMMAND+E	Enter selection into search box
COMMAND+G	Find Again
COMMAND+EQUAL	Replace
COMMAND+H	Replace and Find Again
COMMAND+COMMA	Go To Line
COMMAND+R	Run
COMMAND+PERIOD	Stop
COMMAND+0	Console window becomes visible
PAGE UP	Scroll upward one page
PAGE DOWN	Scroll downward one page
HOME	Scroll to beginning
END	Scroll to end
DEL	Delete character after insertion point
DELETE	Delete character before insertion point
UP ARROW	Move insertion point upward one line
DOWN ARROW	Move insertion point downward one line
RIGHT ARROW	Move insertion point to the right one character
LEFT ARROW	Move insertion point to the left one character

Section 2: Using VeriWell from the Command Line

Introduction

This section describes some basic techniques for getting started with VeriWell from the command line. VeriWell is an interpreted simulator. This means that when VeriWell starts, it reads in the source models, compiles them into internal data structures, and then executes them from these structures. The structures contain enough information that the source can be reproduced from the structures (minus formatting and comments.)

After the model is running, the simulation can be interrupted at any time by pressing CTRL+C. This puts the simulator in an interactive command mode. From here VeriWell commands and Verilog statements can be entered to debug, modify, or control the course of the simulation.

Invoking VeriWell

VeriWell is invoked with one or more source files and zero or more options. An example of a minimal command line is:

```
veriwell model.v
```

This invokes VeriWell, and immediately starts executing the source file **model.v**.

Note: By convention, Verilog source files use a **.v** suffix.

If there is no user intervention, the model will run to completion. If there is more than one file, then each needs to be specified on the command line:

```
veriwell cpu.v memory.v io.v
```

The order that the files are displayed in the command line is the order that the files will be compiled. In most cases the order is irrelevant, but there are some cases where the order is significant, particularly when using the same macros (**'define**) across files.

Frequently, it is desirable to cause VeriWell to enter interactive mode before execution begins. This is accomplished by inserting the **-s** (stop) option:

```
veriwell -s cpu.v memory.v io.v
```

Another frequent command line option is **-t** which enables trace mode at the beginning of the simulation:

```
veriwell -t cpu.v memory.v io.v
```

Command line options may be displayed in any order and anywhere on the command line. The following examples are identical:

```
veriwell -t -s cpu.v memory.v io.v
```

```
veriwell cpu.v memory.v io.v -s -t
```

```
veriwell cpu.v -t memory.v -s io.v
veriwell cpu.v memory.v -s io.v -t
```

Options are processed in the order that they are displayed on the command line. Files are processed in the order that they are displayed after the options are processed.

After invoked, the simulation can be controlled with simple commands. Also, VeriWell accepts any Verilog statement (but new modules or declarations cannot be added.)

- c Compile Only Option

  This option compiles the source text only and then exits. This option can be used for checking for errors without simulating the model.

- f Command Argument File Option

  This option instructs VeriWell to read command options from a text file. Often, there are many files and command line options that make command lines unwieldy. As an alternative to specifying commands and options on the command line, they instead can be contained within a file. This file can contain source text file names and VeriWell command options, including other **-f** options. The **-f** option must be alone in a command line argument and would be displayed as:

veriwell -f command.vc

This gets all source file names and options from the file called **command.vc**.

Note: By convention, command file names end with the **.vc** suffix.

The command file would contain something like this:

```
cpu.v
memory.v
io.v
-s
-t
```

Command files can be combined with command line arguments. Command files can also be nested; that is, a command file can contain a reference to another command file. Source files and options are processed as they are encountered.

- i Input File Option

Syntax: -i <file name>

This option permits interactive commands to be read from a file, where the file name is specified immediately following the **-i** option. Commands can be put into a file and automatically submitted to VeriWell just before the simulation begins. This is accomplished with the command line option. Any command that is legal in interactive mode can also be sent via an input file. You must include the **-s** option with the **-i** option on the command line, or include the **$stop** system task in the input file.

For example, assume the file foo.vi contains the following:

```
$settrace;
#1000 $stop;
```

This file is submitted with the following command:

```
veriwell -i foo.vi cpu.v ...
```

-k Key File Option

Syntax: -k <key file name> | nokey (key name)

This changes the default name of the key file, which retains a log of all keystrokes entered during the simulation run. Specifying a **nokey** disables the key file. By default, the key file is called **veriwell.key**. The key file can be changed, enabled, and disabled, at run time using the **$key** and **$nokey** system tasks.

- l Log File Option

Syntax: -l <log file name> | nolog (log name)

This changes the default name of the log file, to which all output is copied. Specifying **nolog** disables the log file. By default, the log file is called **veriwell.log**. The log file can be changed, enabled, and disabled at run time using the **$log** and **$nolog** system tasks.

- p Password Option

Syntax: -p<passwd>

This option is for versions of VeriWell for Sparc that require a password on the command line (or in a command file). The password is an 8-digit hex number supplied with the distribution.

Note: **There are** no spaces **between the** -p **and the number** <passwd>**.**

- s Stop Option

  This option cause the interactive mode to be entered before the simulation begins. You can also use the **$stop** system task in the input file.

- t Trace Option

  This option performs a full trace from the start of simulation. You could also use the **$settrace** system task in the input file. Trace mode may be disabled with the **$cleartrace** system task.

- y Library Directory Option

Syntax: -y <directory> [library directory]

This option specifies the path of a directory where VeriWell will search for modules not defined in the file list. This is used to implement libraries. If this option is specified, then any undefined modules found during the compiling of the model will be searched in the given directory. The name of the file must be the same as the name of the module. The suffix is determined by the **+libext** option.

Syntax: -v <filename> [library directory]

This option specifies the name of a library file. If a module is not explicilty specified, one of more library files will be searched in order to find the module.

Section 3: Using the VeriWell Simulator

Interactive Control

VeriWell permits commands and statements to be entered interactively. This can be used to control, observe, and debug the simulation.

There are three ways to enter into simulation mode:

- When **CTRL+C** is pressed (or the **stop** button is selected in a GUI) while the simulation is running
- When the **$stop** system task is executed
- When the **-s** command line option is used

When interactive mode is entered, the command prompt will look like this:

```
C1>
```

Where **C** refers to the fact that VeriWell is in command mode, and **1** refers to a sequential number of the completed command.

Interactive Commands and Debugging Techniques

- There are a number of interactive commands that can be entered at the command prompt:
- The **continue** command is produced by entering the period (**.**) character. VeriWell switches from halt mode to run mode and continues the simulation run.
- The **step** command is produced by entering the semicolon (**;**) character. This causes VeriWell to execute the next statement in the source text file and immediately return control back to the user.
- The **trace-step** command is produced by entering the comma (**,**) character. This command performs the same function as the step command, but also generates a trace message from the statement executed.
- The **current location** command or **where** command is produced by entering the colon (:) character. This command will display the current location.

Typing commands interactively can be used for controlling, debugging, and monitoring the simulation. There are probably unlimited ways to use commands interactively, but here are some useful examples:

- Terminating a simulation

A simulation can be ended by typing **$finish;** at the command line.

Note: All Verilog commands require a terminating semicolon, that semicolon must also be typed at the command line:

```
C1> $finish;
```

A simulation can be terminated in a number of other ways such as using **CTRL+C**, choosing **Quit** from the **File** menu in a GUI version, and permitting the simulation to run to completion.

- Enabling and disabling trace

One of the best ways to debug a model is to watch it execute. In VeriWell, there is a trace mode that displays each statements as it is executed. To enable trace mode, type **$settrace;** at the command line (it may also be executed from inside a model). When the simulation is resumed, each executed statement will be displayed. To disable trace mode, type **$cleartrace;.**

It is often desirable to execute one statement at a time and observe the effects of that statement. Pressing the comma (**,**) character at the command prompt executes a single Verilog statement and displays it.

Because everything that is displayed is also sent to the log file, **veriwell.log** by default, results of tracing can be viewed after the simulation is complete.

- Displaying

Any variable can be displayed to view its current contents. Typing **$display(...);** will show the current value. Make sure that the scope is correct. A common mistake is to view a trace, pause the simulation, and type **$display;** without realizing that the variable may not be in the current scope. In interactive mode, the current scope is set using the **$scope** system task. It does not follow the scope of the simulation. For example,

```
C2> $scope (top.cpu1.iunit);

C3> $display (ireg);
```

The same can be expressed,

```
C2> $display (top.cpu1.iunit.ireg);
```

All the variables in a given scope can be displayed using the **$showvars** system task. **$showvars** also displays the information about when the variable was last modified, specifically, the simulation time, the file name, and the line number of the reference.

- Changing variables

Variables can be modified by using simple assignment statements:

```
C2> foo = 4 * bar;
```

- Timed simulations

A simulation can be set to run for a certain length of simulation time:

```
C2> #1000 $stop;.
```

This sets up a delay of 1000 simulation units. When that has passed, the **$stop** statement will be executed, pausing the simulation. The continue command, which is the period (**.**) character, continues the simulation after the statement has been entered.

- Variable Watches (breakpoints)

Interactive statements can be used to stop the simulation when a particular variable, or combination of variables, changes:

```
C2> @(top.cpu1.iunit.ireg) $stop;.
```

This will continue the simulation until the variable changes. However, the statement will not execute immediately after the variable changes. Rather, it will be executed sometime later in the same time unit.

- Test modules

One technique for testing models is to write a test module that communicates with the model under test. VeriWell permits more than one top-level module. One top-level module would serve as the top level for the model under test. A second top-level module can be used as a test bed or monitor module that references signals in the other module using hierarchical names. This is a non-intrusive technique that requires no change to the original model.

Predefined Plus Options

- +maxdelays/+mindelays/+typdelays

Specifies which delay should be used in the **min:typ:max** expressions.

- +define+<macro name>+<macros name> ...

Defines macro names from the command line, generally for use with conditional compilation directives. Any number of macros can be defined.

- +synopsys

Displays warnings for constructs that are either not supported or ignored by Synopsys HDL Compiler.

- +noshow_var_change

By default, VeriWell keeps track of the location and simulation time of where variables are last written. This is displayed in **$showvars**. This feature may cause a slight performance degradation, so it can be disabled with this option.

- +libext+<ext>+<ext> ...

Specifies the file name extension used when searching for libraries in the library directory. This is most often used with the **-y** option. For example:

veriwell cpu.v -y /design/libs +libext+.vl+.vv

This will search the directory /design/libs for libraries whose file name ends with **.vl** and **.vv**.

Note: In VeriWell for MS-DOS and VeriWell for Windows, the slashes are reversed.

- +incdir+<directory1>+<directory2>+...

Specifies the directories that VeriWell searches for included files.

Note: All characters between the pluses are used in the directory name.

Compilation

During the compilation stage of VeriWell, the three phases of the process are displayed to show the progress of the compilation.

Phase 1: The files are read and converted into an internal data structure. Syntax errors and semantic errors regarding undeclared variables or illegal use of variables are reported in this phase.

Phase 2: The model hierarchy is built, module ports are connected, and storage for variables is allocated in this phase. If any module is instantiated more than once, its structure is copied as many times as needed in this phase. Also, module parameters are propagated. Errors reported in this phase deal with missing modules, irregularities of the parameters, and out-of-memory errors during the allocation.

Phase 3: The entire structure is reparsed during which time forward references to tasks and functions are resolved, hierarchical names are resolved, and expression sizes are determined. Errors detected in this phase include semantic errors dealing with hierarchical references that could not be detected in Phase 1, illegal references to functions and tasks, port size discrepancies, and illegal expression sizes.

Note: Most memory is allocated in the first two phases of the compilation.

Compiler Directives

There are various compiler directives that are provided for controlling what happens when VeriWell compiles. All compiler directives are preceded by the accent grave (**‘**) character.

- ‘define

This directive permits you to create macros for text substitution and can be used anywhere within the source text.

- ‘ifdef, ‘else, ‘endif

These are conditional compilation directives. They permit you to select which pieces of source code to compile based on macro definition.

- ‘include

This directive permits you to insert the entire contents of a source file into another file during compilation.

- ‘timescale

This directive specifies the time unit and precision of the simulation. The time unit is the unit of measurement for time values such as the delay values and simulation time. The time precision specifies how VeriWell controls time values.

User-Defined Primitives and Memory Usage

User-Defined Primitives (UDPs) are used to define combinatorial primitives and two-state devices. In VeriWell, the UDP is implemented to be optimized for performance. This is accomplished by creating a table in memory for each UDP definition. Only one such table is used for each UDP definition; every instance of the definition uses the same table.

The table size is very large when the number of inputs exceeds 6. For this reason, the maximum number of inputs is 10 (9 for state UDPs). The maximum table size is approximately 256K.

Specify Blocks

Specify Blocks are used to define pin-to-pin timings and setup-and-hold checks. In VeriWell, Specify Blocks work just as described in the LRM. However, there are some differences.

In VeriWell, there is no concept of expanded nets. Nets that are defined as vectors are not split into individual nets and cannot have their own timing information. Therefore, certain combinations of timing specifications will be ignored. Specifically, there are two ways to describe module paths. One is the so called parallel case (=>) and the other is the full case (*>). In VeriWell, both cases are treated as if they were defined as the parallel case. This does not pertain to scalar nets.

VeriWell supports all of the defined setup and hold systems tasks.

Section 4: VeriWell Implementation Notes

Except for the following implementations, VeriWell behaves exactly as specified by the IEEE-1364 LRM and Verilog-XL.

Port Collapsing

In some implementations of Verilog, if two nets are connected together via a port, the port is **collapsed**, that is, combined into one net. In VeriWell, module ports are connected using transparent continuous assignments. If a register is connected to a net, then the port propagation does not occur immediately when the port changes; rather it is scheduled for later in the same simulation time. But, when a net is connected to a net, then a collapsed port is emulated by forcing the propagation to occur instantly. The effect of this implementation is transparent to the functionality of the model being simulated, but becomes visible during trace.

Port Connections of Different Net Types

VeriWell does not check for the legality of connecting different net types through the hierarchy. For example, if a parent module instantiates a child module, and the net on the parent’s side of a port is a **tri1** while the net on the child’s side of a port is a **tri0**, an oscillation will result.

To use **tri1**, **tri0**, **triand**, and **trior** as ports effectively in VeriWell, they should be declared only in the top-most level in the hierarchy. All lower-level connections should be declared as **wire** or **tri**.

Pullup/Pulldown Workaround

When modeling an open-collector bus, a common technique is to have a **pullup** or **pulldown** gate drive a **wire** net and have drivers pull the bus in the opposite direction with a greater strength when asserting a signal. In VeriWell, drive strengths are not implemented, therefore, this technique will generate an unknown (X) value when a driver attempts to drive a signal in the opposite direction as the pull.

The preferred method for modeling open-collector buses is to use the **triand** or **trior** nets for pullup and pulldown buses, respectively. This net type should only appear in the highest level of the hierarchy in which the bus exists.

Using Trace

Trace is an indispensable tool for debugging Verilog programs. It displays each statement as it is being executed. Depending on the statement, the results are also displayed.

There are three ways to enable trace. One is to specify the **-t** option at the command line. Another is to execute the system task **$settrace** from either the program or from the interactive command line. Also, a single statement will be executed and traced by entering a comma (,) character at the interactive command line. (Multiple commas may also be entered which executes the respective number of statements.)

If a model uses continuous assignments or ports, VeriWell displays the activation of these as part of the trace as soon as the activation occurs. For example, given the continuous assignment, **assign foo = bar;**, when bar changes, the continuous assignment is executed immediately, and this is displayed in the trace. The continuous assignment represents one of possibly many drivers to the net, **foo**; the net itself is scheduled for updating for sometime later in the current simulation time unit.

Because port connections are implemented as continuous assignments, it may take several steps for a signal to propagate from an output port to an input port, especially in cases where there are several ports connected to a net. Trace shows part of this propagation. Signals emanating from an output port travel upward to its parent module; it then travels back down to other connected ports. Each time a signal reaches a new port, the net connected to that port is evaluated and the results are displayed in the trace.

Predefined Macro __VERIWELL__

The macro, **__VERIWELL__**, is predefined so that statements such as:

‘ifdef __VERIWELL__

can be used for VeriWell-specific code, such as for waveform display.

Simulation Statistics

The non-standard system task, **$showstats**, displays statistics about the current simulation, including the amount of memory used and available. Some of the information is provided for diagnostic purposes only.

Displaying Location of Last Value Change

The **$showvars** system task optionally displays the current location in the module and the simulation time the module variables were last changed. This information is updated even if the value did not change (that is the new data is the same as the old data).

Tracking this update information may affect the performance of the simulation slightly. If this is a problem, this feature can be disabled with the **+noshow_var_change** command line option.

User Interrupt

Pressing **CTRL+C**, **COMMAND+C**, or **CTRL+BREAK** (in MS-DOS) during simulation will put VeriWell into interactive mode. Pressing any of these during compilation will halt the compilation and exit to the operating system.

Appendix A: IEEE-1364 LRM Cross Reference

This appendix is a cross-reference into the IEEE-1364 Language Reference Manual (LRM 1.0), in which all differences between the LRM and the Verilog implementation of VeriWell are noted.

Items in **bold** in this section are implemented in VeriWell. If no note appears after a section title, then VeriWell implements that aspect of Verilog verbatim. If a note does appear, then it either documents an implementation-dependent aspect or it describes a restriction. A restriction is preceded by one asterisk (*) which indicates that the restriction will be lifted in a planned enhancement of VeriWell.

1 **Introduction**

2 Lexical Conventions

2.1 Operators

2.2 White Space and Comments

2.3 Numbers

- Sized decimal numbers (e.g. 5'd31) limited to 32 bits; Machine size is 32 bits

2.4 Strings

2.5 Identifiers, Keywords, and System Names

(Identifiers are limited to 128 characters)

3 Data Types

3.1 Value Set

3.2 Registers and Nets

- 3.2.3 Declaration Syntax: Scalared, Vectored, charge strength, and drive strength are not implemented

3.3 Vectors

3.3.1 Specifying Vectors: Maximum vector length is 262,080 bits

- 3.3.2 Vector Net Accessibility: Vectored and scalared key words are not implemented

3.4 Strengths

- Not supported

3.5 Implicit Declarations

3.6 Net Initialization

3.7 Net Types

- 3.7.3 trireg Net: Not implemented
    - 3.7.5 supply Nets: Not implemented

3.8 Memories

3.9 Integers and Times

3.10 Real Numbers

3.11 Parameters

(Range specifications are permited)

4 Expressions

4.1 Operators

4.2 Operands

4.3 Minimum, Typical, Maximum Delay Expressions

4.4 Expression Bit Lengths

5 Assignments

5.1 **Continuous Assignments**

- Scalared, Vectored, Charge Strength, and drive strength are not implemented

5.2 Procedural Assignments

5.3 Accelerated Continuous Assignments

6 Gate and Switch Level Modeling

6.1 Gate and Switch Declaration Syntax

- Strengths not implemented

6.2 and, nand, nor, or, xor, and xnor Gates

6.3 buf and not Gates

6.4 bufif1, bufif0, notif1, and notif0 Gates

6.5 MOS Switches

6.6 Bidirectional Pass Switches

6.7 cmos Gates

6.8 pullup and pulldown Sources

- Not implemented

6.9 Implicit Net Declarations

6.10 Logic Strength Modeling

- Not implemented

6.11 Strengths and Values of Combined Strengths

- Not implemented

6.12 Strength Reduction by Non-Resistive Devices

- Not implemented

6.13 Strength Reduction by Resistive Devices

- Not implemented

6.14 Strength of Net Types

- Not implemented

6.15 Gate and Net Delays

- 6.15.2 trireg Net Charge Delay: Not implemented

7 User-Defined Primitives (UDPs)

7.1 Memory Usage and Performance Considerations

7.2 Syntax

7.3 UDP Definition

7.4 Combinational UDPs

7.5 Level-Sensitive Sequential UDPs

7.6 Edge-Sensitive UDPs

7.7 Sequential UDP Initialization

7.8 UDP Instances

7.9 Compilation

7.10 Symbols to Enhance Readability

7.11 Mixing Level-Sensitive and Edge-Sensitive Descriptions

7.12 Reducing Pessimism

7.13 Level-Sensitive Dominance

7.14 Processing of Simultaneous Input Changes

7.15 Summary of Symbols

7.16 Examples

8 Behavioral Modeling

8.1 Behavioral Model Overview

8.2 Procedural Assignments

8.3 Conditional Statement

8.4 Case Statement

8.5 Looping Statements

8.6 Procedural Timing Controls

- 8.6.6 Intra-Assignment Timing Control: “repeat (x)@(...)” not implemented

8.7 Block Statements

8.8 Structured Procedures

9 Tasks and Functions

9.1 Distinctions between Tasks and Functions

9.2 Tasks and Task Enabling

9.3 Functions and Function Calling

10 Disabling of Named Blocks and Tasks

11 Procedural Continuous Assignments

12 Hierarchical Structures

12.1 Modules

12.2 Overriding Module Parameter Values

12.3 Macro Modules

- Macro Modules are not implemented

12.4 Ports

12.4.6 Port Collapsing: See Implementation Notes

12.4.7 Port Connection Rules: See Implementation Notes

12.5 Hierarchical Names

12.5.3 Upward Name Referencing: See Implementation Notes

12.6 Scope Rules

13 Specify Blocks

13.1 Declaring Parameters in Specify Blocks

13.2 Module Path Delays

- 13.2.1 Describing Module Paths: Only full paths are supported; using parallel

  paths will generate a warning

- 13.2.13 Global Path Pulse Control: Not supported

- 13.2.14 Pulse Control for Specific Modules and Module Paths: Not supported

13.3 Timing Checks

13.4 Using Path Delays and Timing Checks in Behavioral Descriptions

13.5 **Interactive Back Annotation**

A **Formal Syntax Definition**
Subset as outlined previously

B System Tasks and Functions

B.1 The Display and Write Tasks

B.1.2 Format Specifications: %h, %o, %d, %b, %c, %m, %x, %t, %s, %f, %e, %g are implemented

- B.1.5 Strength Format: Not applicable since strengths are not implemented

B.2 Strobed Monitoring

B.3 Continuous Monitoring

B.4 Timescale Systems Functions

B.5 Timescale System Tasks

B.6 Simulation Time—The $time Function

B.7 Finish System Task

B.8 Functions and Tasks for Reals

B.9 Timing Checks

C Compiler Directives

C.1 ‘define

C.2 ‘default_nettype

- Default net type is always Wire

C.3 ‘unconnected_drive and ‘nounconnected_drive

- Not implemented

C.4 ‘resetall

- Not implemented

C.5 ‘timescale

D List of System Task and System Function Keywords

D.1 $bitstoreal

D.2 $countdrivers

- Not implemented

D.3 $display

D.4 Value Change Dump File Tasks

D.5 File Output

D.6 Finish

D.7 $getpattern

- Not implemented

D.8 $history

- Not implemented

D.9 $incsave

- Not implemented

D.10 $input

D.11 **$itor**

D.12 $key and $nokey

D.13 $list

- Not implemented

D.14 $log and $nolog

D.15 $monitor, $monitoron, $monitoroff

D.16 $printtimescale

D.17 $readmemb and $readmemh

D.18 $realtime

D.19 $realtobits

D.20 $reset, $reset_count, $reset_value

- Not implemented

D.21 $restart

- Not implemented

D.22 $rtoi

D.23 $Saving and Restarting

- Not implemented

D.24 $scale

D.25 $scope

D.26 $showscopes

D.27 $showvars

Takes zero or one argument; does not display driver information

D.29 $stime

D.30 $stop

D.31 $strobe

D.32 $time, $stime, $realtime

D.33 $timeformat

D.34 $write

E List of Compiler Directive Keywords

E.5 ‘define

E.7 ‘ifdef, ‘else, ‘endif

E.8 ‘include

E.9 ‘timescale

Appendix B: Implementation Differences from Verilog-XL

Introduction

This appendix describes the differences between the way VeriWell works and the way Verilog-XL works. Note that these differences are subtle and will not affect the execution of well-written Verilog models.

Event Ordering

The order that events are scheduled and executed is consistent with Verilog-XL to the extent possible. The reason for doing this is not so that models are guaranteed to work under both VeriWell and Verilog-XL, rather, VeriWell was designed such that users can trace models in VeriWell and in Verilog-XL with little noticeable difference. However, it should be noted that models that depend on the order of execution are considered to be not well-written because they reflect race conditions and may perform unpredictably in other vendors' Verilog, or even in future releases of VeriWell (or Verilog-XL).

In some cases, the order of net scheduling may be different. This is because Verilog-XL schedules nets differently depending on the type of net, whether it is sourced by a continuing assignment, and net assignment, or a port, and whether a port is collapsed. In most cases, net scheduling will track that of Verilog-XL.

Module Ports and Port Collapsing

Port connections are implemented as continuous assignments in VeriWell. Rules for port connections are similar to those of Verilog-XL, but there are some differences. In Verilog-XL, under certain circumstances, ports are **collapsed**, that is, if each side is a net, then one of the nets disappears and only one is used. This is a performance enhancement.

VeriWell emulates port collapsing by immediately propagating values across ports that have been **collapsed**. This is unlike Verilog-XL, which actually combines nets that have been collapsed. Verilog-XL will expand vector nets into arrays of scalar nets if a port connects two different sized nets, or if one or both sides are concatenations or part selects. VeriWell does not implement expansion of nets, so it could not handle these cases with building continuous assignments.

VeriWell will **collapse** a port if both sides of a port are scalar nets or if both sides are vector nets. Therefore, there are some cases when VeriWell will not collapse a port, but where Verilog-XL will. This may cause a disparity in the way nets are scheduled in the two simulators.

Control Expressions Limited to 32 Bits

Expressions used by VeriWell for control are limited to 32 bits. This includes repeat counts, delay values, part- and bit-select and array index expressions, and shift counts. A compile-time error will result if the expression attempts to evaluate a number greater than 32 bits.

$Monitor

Unlike Verilog-XL, the $monitor statement will be triggered if any variable in the argument list changes. In Verilog-XL, $monitor changes only when and argument expression changes. For example, the statement:

$monitor (a + b);

will not be triggered if both a and b changes, but the sum stays the same. In VeriWell, the statement will be triggered in this case.

Scoping

VeriWell uses a different technique of storing variables than Verilog-XL. Variables in models are handled the same, but in interactive mode, variables can be accessed in parent modules without scoping, unless, of course, that variable has been redefined in a lower scope. For example, if a register **a_reg** is defined in a top-level module and **$scope** points to some lower-level module, typing **$display (a_reg)** will execute legally in VeriWell, but not in Verilog-XL.

Key File

The key file in VeriWell will capture CTRL+C, but this cannot be fed back into VeriWell as input file because there is no information on where in the simulation the CTRL+C occurred.