VeriWell User's Guide

Wellspring Solutions, Inc.
P.O. Box 150
Sutton, MA 01590
(508) 865-7271 - fax: (508) 865-1113
info@wellspring.com

Table of Contents

Before You Begin

This chapter provides VeriWell users with important and useful information.

Copyright

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' software products, is found at the end of this document. Customers are expected to read this agreement before breaking the seal on the package containing the software.

Registration

We advise customers to register their software by filling out the registration form found with the printed manual and mailing or faxing it as indicated. Because many copies of the software are sold to end users through agents and distributors, Wellspring Solutions will not have you in its records until you do register. Once registered, we can include you in our records and keep you informed about current developments at Wellspring Solutions and the availability of new revisions.

Customer Support

For technical assistance, customers can call: (508) 865-7271 between 9am and 5pm EST, fax: (508) 865-1113, Email: info@wellspring.com, or write: P.O. Box 150, Sutton, MA 01590, USA. Technical problems can also be reported using the Wellspring Bulletin Board System (BBS).

Ordering

Wellspring Solutions products can be ordered directly in the U.S. by calling the toll-free number 1-800-VERIWELL (1-800-837-4935), 9am-5pm EST. For orders outside the U.S., call (508) 865-7271. In Europe, contact Acapella, Ltd., +44-703-769-008. In Japan, contact HY Associates at +81-3-3929-7111, or Advanced Control Technology at +81-426-44-5308.

Info-VeriWell Electronic Mail List

Email is the most efficient communication method to report problems and inform users of availability of updates and new products. Email also allows users to have their own autonomous VeriWell User's Group. If you are on the Internet (or subscribe to Compuserve or America Online), send mail to:

Info-VeriWell-request@DMC.com

Put the word 'subscribe' in the body of the mail and information about the mailing list will be sent to you.

Receiving Updates

Wellspring Solutions regularly improves current revisions of the software. These updates are free and can be obtained in a relatively open manner -- by the Bulletin Board System, FTP, Email, or mail. Once you register, we can keep you informed of the status of BBS and FTP. Readers of the Info-VeriWell electronic mail list will also be informed about the availability of upgrades.

Users without Email access should call the BBS periodically to check on the availability of updates. If you can't access BBS, FTP, or Email, please call our main number to request the update be sent by regular mail.

Bulletin Board System (BBS)

Wellspring Solutions maintains an electronic Bulletin Board System (BBS) that is available by modem 24 hours per day 7 days per week. The BBS is always the first to receive new products, updates, and bug fixes. Beta versions and custom versions are also on the BBS (in passworded areas). If you do not have Email access, you can send messages and upload code fragments to the BBS for problem diagnosis.

The BBS is divided into two sections: a section for uploading and downloading files and a section for sending and receiving messages (mail). Both sections are further divided into areas for particular product and interest categories. The BBS is menu driven, and usage is self-explanatory.

To use the BBS, you must have:

A **Modem**

If you are planning to download software, we strongly recommend that you use a 9600-baud modem or faster (v.32 or v.32bis).

Communications Software

Your communications software should support the Zmodem transfer protocol. Most files are approximately 300kb in size and will take a few minutes to download at 9600 baud with Zmodem. The BBS also supports Xmodem and Ymodem.

We suggest that you familiarize youself with the uploading and downloading procedures of your communications software.

To log onto the BBS:

Use the communication software to dial 1-508-865-1113 (yes, that is also our fax number)

Make sure that your modem is set to 8-N-1 (8 bits, no parity, one stop bit)

Identify yourself and create a password

Follow the menus. The BBS is divided into File areas and Message areas. Use the menu selections to go to one of the File areas to download or upload files or to one of the Message areas to send and receive messages (mail).

Note: The quality of the telephone lines can affect your ability to connect and transfer data at high speeds.

FTP

Many files and updates are available over the Internet via the File Transfer Protocol (FTP) utility. The files are located on <dmc.com>. Please note that this is a VAX system, so if you are accustomed to using FTP on a Unix system, there are a few minor differences. To access the files on DMC, use the following procedure:

```
ftp dmc.com
login: anonymous
password: <your email address>
cd [000000]   (This is the root directory)
cd wellspring
cd v386    (or whatever directory is of interest)
binary
hash
get veri12z.zip   (or whichever file is of interest)
quit
```

Additionally, the commands "ls" and "dir" can be used to display the contents of each directory. For more information, consult the FTP reference or ask your system administrator.

Email

We will send updates to customers by Email, however, this method is used only on a need basis because it presents a few complications. *First*, software sent by Email must be encoded into ASCII because only ASCII files can be transmitted by Email over the Internet. *Secondly*, software is generally too large to be transmitted as one message (ie., VeriWell), therefore, it must be sent as several messages.

We use *uuencode* to convert software into ASCII format. *Uuencode* is usually included in the distribution of UNIX systems. Free *uuencode* versions are available for MSDOS. If needed, Wellspring will provide a version of *uuencode* written in BASIC to convert VeriWell upgrades to ASCII format. Please call our main number for more information.

Mail

Updates can also be sent on distribution media by regular mail. If you're not able to access any of the electronic methods described, or these methods are not convenient for you, call our main number and arrangements can be made to mail you updates.

Publications

We recommend that users have a copy of the *Verilog HDL Language Reference Manual (LRM)*. Inquiries related to purchasing the ***LRM,*** or requests to be placed on the OVI Newsletter mailing list can be directed to Open Verilog International (OVI), 15466 Los Gatos Boulevard, Suite 109-071, Los Gatos, CA 95032; Tel: (408) 353-8899, Fax: (408) 353-8869, Email: OVI@netcom.com.

Wellspring Solutions also recommends two published books on Verilog:

The Verilog Hardware Description Language, by Thomas and Moody, published by Kluwer Academic Publishers, Norwell MA, ISBN 0-7923-9126-8

Digital Design with Verilog HDL, by Sternheim, Singh, and Trivedi, published by Automata Publishing Co., Cupertino, CA, ISBN 0-9627488-0-3

Preface

About Verilog HDL

Verilog, the EDA industry's first standard HDL, was introduced in 1985 by Gateway Design System Corporation, now a part of Cadence Design Systems, Inc.'s Systems Division. Until May, 1990, with the formation of Open Verilog International (OVI), Verilog HDL was a proprietary language of Cadence. Cadence was motivated to open the language to the Public Domain with the expectation that the market for Verilog HDL-related software products would grow more rapidly with broader acceptance of the language. Cadence realized that Verilog HDL users wanted other EDA software and services companies to embrace the language and develop Verilog-supported design tools.

HDLs allow designers to describe designs at higher levels of abstraction, such as architectural or behavioral, and provide a path to logic synthesis. Verilog HDL also allows for mixed-level designs, where users can describe a design at both high and low levels of logic simulation and synthesis. Designers are choosing top-down design and mixed-level design to contend with ever-increasing complexities and shrinking time to market cycles.

About VeriWell

VeriWell is a comprehensive implementation of Verilog HDL. VeriWell supports a number of platforms and operating environments. These currently include 386/486/Pentium systems under DOS, Sparc or Sparc compatible systems under SunOS 4.1.x or greater and Solaris. VeriWell is designed to be as portable as possible. Nearly 100% of the sources are shared between the different platform versions. The DOS version uses a DOS extender to compensate for the shortcomings of DOS and to fully utilize the 32-bit architecture of the 386/486/Pentium processors.

VeriWell supports the Verilog language as specified by the OVI Language Reference Manual. 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 OVI 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 IC designers and consultants for all pre-synthesis model development. As new features are added, VeriWell can be used in all phases of model development, including structural verification and back-annotated timing verification. As a component of a large-scale top-down design methodology, VeriWell is used in conjunction with other high-end OVI-compliant simulators, such as Verilog-XL or Chronologic's VCS, to develop behavioral, RTL and synthesis models in Verilog HDL, greatly reducing 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 finds most syntax and semantic errors on the first pass. This helps end the frustration of users who hitherto would fix all reported errors only to find a new set of errors reported.

Wellspring Solutions is committed to providing a comprehensive, compatible, and affordable design tool for Verilog users. To that end, Wellspring Solutions will continue to upgrade and enhance the basic VeriWell package and offer customers quality and timely technical support.

Scope of this Document

The purpose of this document is to describe where VeriWell differs from both the LRM and Verilog-XL. This document is neither a tutorial nor a reference. It assumes knowledge of the Verilog Hardware Description Language (HDL). New users are encouraged to read one of the recommended books about Verilog.[1] Experienced XL users are advised to read the sections in this document detailing the differences between VeriWell and XL. All VeriWell users should obtain the LRM from OVI since that is the definitive reference on which VeriWell is based. XL users can use the manuals from Cadence, although there may be differences that are not documented in the VeriWell manual.

Divisions in this Document

Before You Begin includes information about the Wellspring Solutions, Inc. copyright and software license agreement, ordering procedures, product registration, technical support, on-line communication, updates, and recommended publications.

Preface: This section; presents a brief history of the Verilog language, describes the VeriWell simulator, and defines the scope of this guide.

Part I, VeriWell, covers all ports of VeriWell. Since all of the language aspects are common to all ports, this section is written to be platform independent. **Part I** describes how to use VeriWell and also describes VeriWell with regard to the OVI *Language Reference Manual* and Verilog-XL:

Chapter 1: *Using VeriWell* explains how to get started and run VeriWell, lists commands and options, and describes the stages of compilation.

Chapter 2: The *VeriWell Feature List* is a summary of all the language constructs supported by VeriWell.

Chapter 3: The *OVI LRM Cross-Reference* describes how VeriWell deviates from the LRM by listing each section of the LRM and describing all applicable deviations.

Chapter 4: *VeriWell Implementation Notes* describes certain implementation elements of VeriWell that are not covered or would not be applicable in the LRM.

Chapter 5: *Implementation Differences from Verilog-XL* describes differences between the way VeriWell works and the way Verilog-XL works.

Part II, Platform-Dependent Usage, describes operating environment characteristics of the currently supported ports. Those aspects of the VeriWell simulator that are dependent on the platform or environment are described in this section. Such aspects include system requirements, installation procedures (also described in an on-line text document on the distribution media) and other environment-specific particulars.

Chapter 1: *Using VeriWell under DOS*

Chapter 2: *Using VeriWell under UNIX*

This guide is not very long. VeriWell is designed to be almost idiosyncrasy-for-idiosyncrasy compatible with Verilog-XL, therefore, there are few differences to note. Ideally, a two-page installation guide would be all that is necessary, but the fact is, there will always be some differences. We will continue to minimize and document those differences.

[1]Reference *Before You Begin: Publications*

Part I: VeriWell

1 Using VeriWell

VeriWell is run from the command line. Type "veriwell" followed by the names of the files containing the models and the options. The options can appear in any order and anywhere on the command line. For example:

```
veriwell cpu.v bus.v top.v -s
```

This will load each of the files into memory, compile them, and enter interactive mode. Removing the "-s" option would cause the simulation to begin immediately.

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

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

Interactive mode is entered in one of three ways:

When the "-s" option is used on the command line (or in a command file), interactive mode is entered before the simulation begins,

When the simulation encounters the $stop system task, or,

When the user types CTL-C (or CTL-BREAK in DOS) during simulation (but not during compilation).

Interactive Commands

Continue ('.') [period]

Resume execution from the current location.

Single-step with trace (',') [comma]

Execute a single statement and display the trace for that statement.

Single-step without trace (';') [semicolon]

Execute a single statement without trace.

Current location (':') [colon]

Display the current location.

Typically, the kinds of Verilog statements executed interactively are used for debugging and information-gathering. `$display` and `$showvars` can be typed at the interactive prompt to show the values of variables. `$scope` and `$showscopes` can be typed to traverse the model hierarchy. `$settrace` and `$cleartrace` will enter and exit trace mode. Typing "`#100; $stop;`" will stop the execution after 100 simulation units.

Commandline Options

-c (compile only)

Causes the model to be compiled only and not simulated.

-f <commandfilename> (command file)

Reads additional command line options from a file, including source file names. Each option or source file name is separated by a new line. Comments are allowed in command files. Command files may be nested. This option is generally used to specify the names of the source files so that they do not have to be typed in every time the simulation needs to be rerun. Also, if passwords are applicable (on workstation versions), all passwords can be put into a single command file. For example:

```
veriwell -f command.vc
```

In command.vc:

```
cpu.v
memory.v
bus.v
top.v
-s
```

-i <inputfilename> (input file)

Specifies a file that contains interactive commands to be executed as soon as interactive command mode is entered. This option should be used with the "-s" option. This can be used to initialize variables and set time limits on the simulation.

-s (stop)

Causes interactive mode to be entered before the simulation begins.

-t (trace)

Causes all statements to be traced. Trace mode may be disabled with the $cleartrace system task.

-l <logfilename> | "nolog" (log name)

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.

-k <keyfilename> | "nokey" (key name)

Changes the default name of the key file, which retains a log of all keystrokes entered during the simulation run. Specifying "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 timeusing the $key and $nokey system tasks.

-y <directory> [library directory]

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.

-p<passwd>

This option is for versions of VeriWell 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 that there are <u>no spaces</u> between the -p and the number.

Predefined Plus Options

+maxdelays/+mindelays/+typdelays

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

+define+<macro name>+<macro 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 at compile time 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 filename 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 filename ends with ".vl" and ".vv". (Note that on DOS- and Windows-based systems, the slashes are reversed.)

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

Specifies the directories that VeriWell searches for include files. (Note that 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. These three phases are:

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 (i.e. the most common types of errors) 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. Note that most memory is allocated in the first two phases of the compilation.

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.

2 VeriWell Feature List

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

Data Types

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

Operands (Conforms to Expression Bit-Length Rules)

Static types:

REG
REG arrays
INTEGER
INTEGER arrays
TIME
REAL
MEMORIES
PARAMETER

Nets:

WIRE/TRI
WOR/TRIOR
WIAND/TRIAND
TRI0
TRI1

Not implemented:

SUPPLY0/1
TRIREG

Operators

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

Built-in Primitives

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

Statements

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

Not yet implemented: Repeat event control in intra-assignment events

Hierarchical Structures

Port connections by ordered list
Port collapsing
Named Ports

System Tasks and Functions

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

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".
-y (library directory)

Compiler Directives

`define
`ifdef, `else, `endif
`include

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.)
+encrypt (hardware key versions only)
+noshow_var_change (disables tracking of location and time of each variable update)
+libext (library extension)
+incdir (include directory)

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. [Note that the indices for a vector may be any number up to 4G (the highest number represented in 32 bits), but that the range must not be larger than 262,080.]

3 OVI LRM Cross-Reference

This chapter is a cross-reference into the ***Language Reference Manual*** **(LRM)***,* 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 VeriWell 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, two, or three asterisks which indicate when that restriction will be lifted: In a future version of VeriWell (*), not planned (**), or by a future VeriWell add-on (***).

*	To be enhanced in a later version
**	No enhancement planned
***	Supported as part of an add-on

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 dentifiers, 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 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 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 allowed

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, drive strength not implemented*

5.2 Procedural Assignments

6 Gate and Switch Level Modeling

****Gates are emulated through module instantiations*

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

****Not implemented*

6.6 Bi-directional Pass Switches

6.7 CMOS Gates

****Not implemented*

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)

****2.0 gate module*

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 Structure

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

12.5.1 Upward Name Referencing: see Implementation Notes

12.6 Scope Rules

13 Specify Blocks

****2.0 gate module*

A Formal Syntax Definition

Subset as outlined above

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 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

implemented in 2.0

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

**implemented in 2.0*

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

**Not implemented*

D.17 $readmemb and $readmemh

D.18 $realtime

**Not implemented*

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

**Not implemented*

D.25 $scope

D.26 $showscopes

D.27 $showvars

**Takes zero or one argument; does not display driver information*

D.28 $readmemb and $readmemb

D.29 $stime

D.30 $stop

D.31 $strobe

D.32 $time, $stime, $realtime

**$realtime is not implemented*

D.33 $timeformat

**Not implemented*

D.34 $write

E List of Compiler Directive Keywords

E.5 'define

E.7 'ifdef, 'else, 'endif

E.8 'include

4 VeriWell Implementation Notes

Except for the following implementations, VeriWell behaves exactly as specified by the OVI 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 implemenation 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 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 (Pre 2.0 only)

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.

Gates and Libraries (Pre 2.0 only)

VeriWell supports library lookups and includes a standard set of library modules. The standard library contains RTL-level emulation of the standard Verilog gates: **AND, NAND, OR, NOR, XOR, XNOR, BUF, NOT, BUFIFx, NOTIFx,** etc. The libraries are automatically instantiated if the respective keywords are used in the module instantiation name.

To accommodate the variable number of ports in the logic gates (AND, OR, etc.), a file naming convention is used that indicates the total number of ports in the instantiation. For example, the file "AND.4VL" contains the module for a 3-input and gate (four total ports). The file "XOR.0VL" is a 9-input xor gate. In this way, VeriWell will allow instances of the same gate but of different sizes using the instantiation of modules.

Gate delays are also emulating through parameter overrides. As an extension to Verilog, VeriWell extends the Verilog specifications to support the min/typ/max timing syntax as a parameter override so that models containing instantiation of gates will be compatible with other Verilog simulators.

The default location of the library is in the sub directory "LIB" under the "`VeriWell.exe`" currently under execution. An environment variable, "**`VLIBPATH`**", can be set with a search path to override the default. The library will not be referenced if there is already a similarly named module in the hierarchy.

Specify Blocks (2.0 gate module)

Specify blocks are implemented as per the LRM. However, since VeriWell does not implement vector net expansion, there is a minor limitation: parallel module paths ("=>") are not supported and will automatically convert into full module paths.

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 statement's 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 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.[2]

Since 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.

[2]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.

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 the amount available. Some of the information is provided for diagnostic purposes only.

Displaying Location of Last Value Change

In VeriWell, the **`$showvars`** system task optionally displays the location in the module, as well as the simulation time, of the last time variables were written. This information is updated even if the value did not change (i.e. 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 **Control-C** or **Control-Break** (in DOS) during simulation will put VeriWell into interactive mode. Pressing either during compilation will halt the compilation and exit to the operating system.

New System Tasks

$settrace, $cleartrace

Enables and disables trace mode.

$showstats

Displays information about the amount of memory being consumed by the current model and how VeriWell views the model.

5 Implementation Differences from Verilog-XL

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 since they reflect race conditions and may perform unpredictably in other vendor's Verilog, or even in future releases of the 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. 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 control-C, but this cannot be fed back into VeriWell as input file because there is no information on where in the simulation the control-C occurred.

Part II: Platform-Dependent Usage

1 Using VeriWell under DOS

This chapter contains information about VeriWell that applies only to the DOS version, including system requirements and installation procedures.

Introduction

There are two DOS versions of VeriWell available, the 16-bit version (known as VeriWell/PC) and the 32-bit version (known as VeriWell/386 -- although it is optimized for use on 486-based systems). The 16-bit version is a standard DOS application, is limited by DOS to a maximum of 640k bytes of usable memory, and works on older 8088- and 286-based PCs. The 32-bit version uses a "DOS extender" to put the host processor into "protected mode" which then utilizes the full capabilities of the processor. The 32-bit version supports up to 64MB of physical memory or 32MB for virtual memory. It is also much faster than the 16-bit version.

The 32-bit version is intended as an industrial-strength simulator, where the 16-bit version is intended as an educational tool.

Both versions require the use of a hardware key which is shipped with the manual and disks. The hardware key is plugged into a printer port of the system. A printer can be connected to the other side of the key. The existence of the key is transparent to all printer operations.

If VeriWell is invoked without the key, it will still run, but with certain limitations. The so-called "unregistered" version can be used as an educational tool for learning Verilog or for running small simulations. These limitations are described in a text file that is included with the distribution.

System Requirements

The 32-bit verision of Veriwell requires DOS 5 or later, 2MB of RAM and 2MB of disk space. Up to 32 MB of disk space is needed if virtual memory is enabled in the DOS extender.

What is Included

The contents of the files included with VeriWell are described in your distribution media. VeriWell may be distributed in several ways. If you received the software electronically (BBS, FTP, Email), then, most likely, all of the files are compressed into a single "zip" file. This requires that an "unzipper" be used, such as PKUNZIP version 2.0x, or UNZIP. If you received VeriWell on a floppy disk, then some files are zipped, but the unzip utility is included.

Installation

Attach the hardware key (if provided) to a printer port (it does not matter which; hardware keys can also be connected together if there are more than one).

Create (`mkdir`) and change to (`cd`) a target directory on your hard drive.

Copy all the files from the distribution disk to the target directory.

Run the included `UNZIP` on `EXE.ZIP` and `EXAMPLES.ZIP` (e.g.: `UNZIP EXE.ZIP`)

(Pre-2.0 ony) Create and change to a sub directory called `LIB` beneath the target directory. (The libraries are necessary only if the built-in primitives are to be used. They can be installed later, if desired, and/or in a different directory. Libraries are explained in detail later in this document.)

(Pre-2.0 only) From `LIB`, run "`..\MAKELIB`" (this generates the libraries).

Edit `AUTOEXEC.BAT` to:

include the target directory in the `PATH`

```
(Optional)SET DOS4GVM=1     (This enables virtual memory)
```

Ensure that `FILES` in `CONFIG.SYS` is set to 20 or higher.

(Optional) Create another directory for `BUFFIT30.ZIP`; `UNZIP` it and follow

its directions. For example:

```
cd \
mkdir veriwell
copy a:*.*
unzip exe
unzip examples
mkdir lib
cd lib
..\makelib
cd ..
```

Finally, it is recommended that the distribution disk be copied onto a backup floppy.

Installation is complete and VeriWell is now ready to run.

About the DOS Extender

The 32-bit version of VeriWell is shipped with a DOS extender, DOS/4GW, from Rational Systems, Inc. To run this version of VeriWell, DOS/4GW must be invoked first. DOS/4GW puts the system into protected mode (DOS normally uses REAL -- 8086 emulation -- mode) and, optionally, sets up paging. The on-line file, DOS4GW.DOC, provides detailed information on configuring DOS/4GW through the use of environmental variables.

The executable file, `dos4gw.exe`, expects as its first argument the DOS application that requires it, in this case `veriwell.exe`. Additional command line arguments apply to the `veriwell.exe` application. For example:

```
DOS4GW veriwell cpu.v
```

To simplify matters, `veriwell.exe` can be run without explicitly invoking DOS4GW at the command line. When `veriwell.exe` is run, it will look for and execute `dos4gw.exe` first. For this to work properly, dos4gw.exe must either be in the path, or in the same directory as `veriwell.exe`.

DOS Considerations

There are several aspects of using Verilog under DOS that are not relevant under UNIX. These have to do with memory display and interaction.

Memory

VeriWell uses a DOS extender which gives it the ability to access all the memory in the system. To enable page swapping, set the environment variable, **DOS4GVM**, to 1 (e.g. set `DOS4GVM=1`). For more information and options regarding the memory manager, see the online file on the distribution disk.

Beyond the memory used to compile a Verilog model, little additional memory is used during runtime. Once the model is compiled and is running, the only additional memory that is used is during the processing of interactive statements. Non-blocking assignments with delays use some memory during processing and will return it after completion.

Scrolling and Shelling

DOS does not provide a windowing environment, yet the development of Verilog models is greatly eased in a windowing environment. Specifically, it is often helpful to have an active Verilog window alongside an editor window in which to view the code and make changes when necessary. Also, it is generally desirable to run Verilog in a scrollable window. Both of these issues have been addressed by VeriWell.

VeriWell provides the ability to execute an external command and to shell out to DOS at the VeriWell command line. A single exclamation mark ('!') followed by a command line will execute that command in a DOS shell. For example:

```
"!freemacs model.v".
```

A single exclamation mark by itself will open a DOS shell.

Interactive Usage

The current interactive command environment is based on DOS routines which prevent "hot keys" from being acted upon by DOS. In other words, certain keystrokes are filtered and prevented from being captured by DOS. Specifically, Buffit uses a hot key for enabling scrolling. The hot key (by default, alt-F9), will not work while in VeriWell. Shelling out to DOS will enable the use of hot keys, and will not appreciably lose output.

Likewise, **control-D** will not have an immediate effect under VeriWell, unlike Verilog-XL. After control-D is pressed, a carriage return must follow.

43 and 50 Lines per Screen

Three utilities are provided to produce smaller screen font sizes so that more output is visible at one time. The utilities `43.com` and `50.com` produce screen fonts that allow 43 and 50 lines on the screen, respectively. The utility `25.com` restores the font size to the normal size. Each of the fonts' sizes are the same width, that is, 80 characters per line.

Control-C and Control-Break

DOS senses that a control-C has been pressed only if it is the first character in DOS's keyboard buffer. If another key is pressed, perhaps accidentally, before the control-C is pressed, then DOS does not recognize it. This may give the impression that the system is hung. Pressing control-Break is always detected by DOS, so, **if control-C does not work, always try control-Break.**

Positioning of Caps Lock and Control Keys

Most PC keyboards have the Caps Lock key and the Control key in the opposite position than most UNIX workstations. Seasoned UNIX users may be unaccustomed to the position of these keys on a PC. Several programs are available that reverse the position of these keys on the PC. Wellspring Solutions, Inc. will provide such a program at no cost upon request.

Line Terminators

Text files in DOS use a different line terminator than text files in UNIX. In DOS, lines end with a carriage return/line feed combination. In UNIX, lines end with only the line feed.

When transferring Verilog files (or any other text files) from DOS to UNIX and UNIX to DOS, the carriage return must be stripped or added. This can be done in most editors using global search-and-replace functions.

Also, there are two utilities included for doing this: `dtou.exe` converts a DOS text file to UNIX format and `utod.exe` converts a UNIX text file to DOS format.

Simulation Time Information

The system tasks **`$stop(2)`** and **`$finish(2)`** display time and memory information. The time information is based on wall-clock, rather than actual CPU, time. The clock is paused while waiting for interactive commands, but is not paused for other interruptions, such as hitting control-S.

2 Using VeriWell under UNIX

This chapter explains VeriWell installation instructions on systems using UNIX operation system, defines system requirements, and describes how to use VeriWell under UNIX.

System Requirements

Sparc and Sparc compatible systems using SunOS 4.0.x or greater and Solaris

8MB RAM, 2MB disk space.

What is Included

The contents of the files included with VeriWell are described in your distribution media. VeriWell may be distributed in several ways. If you received the software electronically (BBS, FTP, Email), then, most likely, all of the files are compressed into a single "zip" file. This requires that an "unzipper" be used, such as PKUNZIP version 2.0x, or UNZIP. If you received VeriWell on a floppy disk, then some files are zipped, but the unzip utility is included.

Installation

To install VeriWell under UNIX:

Unarchive the VeriWell distribution disk:

```
% cd <VeriWell-directory>
% tar xvf <disk-device>
```

(Pre 2.0 only) Create the libraries:

```
% cd lib
% makelib
% rm makelib
% setenv VLIBPATH <VeriWell-directory>/lib
```

Add `<VeriWell-directory>` to your `PATH` and add `setenv VLIBPATH` to the `.cshrc` or `.login` file.

Finally, it is recommended that the distribution disk be copied onto a backup floppy.

Using VeriWell under UNIX

To use VeriWell, simply type "`veriwell`" at the UNIX prompt, followed by one or more module filenames containing the model to be simulated. Command line options may appear in any order and anywhere on the command line (after `veriwell`). For example:

```
veriwell cpu.v memory.v bus.v top.v -s
```

This will load each of the files into memory, compile them, and enter interactive mode. Removing the "-s" option would cause the simulation to begin immediately.

Passwords

VeriWell on Sparc is password protected on a node-locked basis. The password is in the form of an 8-bit hex number. The password is unique for each hostid. To invoke VeriWell with the password, type:

```
veriwell -p12345678 ...
```

or

```
veriwell -f <passfile> ...
```

where <passfile> is a file in the form of:

```
-p12345678 // password for hostid 1234cdef
-p9abcdef  // password for hostid cdef1234...
```

The command line can be made less verbose by the use of aliases. For example:

```
alias ver veriwell -f /veriwell/passwd
```

Wellspring Solutions, Inc. Software License Agreement

This 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 disk packet and any other materials that are a part of this product within 10 days to WELLSPRING SOLUTIONS and your money will be refunded. No refunds will be given for products that have an opened disk 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. who grants you a personal, nontransferable and nonexclusive 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

tamper with the hardware key included with the Software

TERM

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

the media on which the Software is distributed and the documentation are free from defects in materials and workmanship.

the software will substantially conform to Wellspring's published specifications and to the documentation provided with it when used as specified in such documentation.

However, WELLSPRING SOLUTIONS 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 shall at its discretion either (a) correct the non-conformance of the Software; or (b) replace the defective Software/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 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 above, the WELLSPRING SOLUTIONS Software Product and printed documentation are provided 'AS IS'. WELLSPRING SOLUTIONS 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 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 Product, even if WELLSPRING SOLUTIONS 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 bear any liability for any reason whatsoever and in no case shall Wellspring'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 of an ultimate destination of the Software, Licensee shall not re-export or transfer the WELLSPRING SOLUTIONS Software to anyone outside the U.S.A. without first obtaining a license from the U.S. Department of Commerce, as required.

If you have any questions about this Agreement, direct them in writing to: Wellspring Solutions, Inc., P.O. Box 150, Sutton, MA 01590.