

VHDL, Verilog and SystemC with Blue Pacific

[return to top](#)

VHDL Verilog C C++ Java

Blue Pacific Computing, Inc. -- San Diego, California
Phone: (858) 484-7500 Fax: (858) 674-1127 Email: info@bluepc.com
Site Map: [Home](#) | [Intro](#) | [Classes](#) | [BlueHDL](#) | [Download](#) | [Links](#)

BlueHDL User's Manual Table of Contents

Version 09/20/01

- [Introduction](#)
- [High-Speed Waveform Display](#)
- [Overview of Tutorials](#)
- [Tutorial 1](#)
- [Tutorial 2](#)
- [Tutorial 3](#)
- [Help](#)
- [A Short History of the *BlueHDL* Tools](#)
- [Tcl/Tk Provides User Customization](#)
- [The HDL Compilers](#)
- [The *BlueSim* Simulator](#)
- [The *BlueWave* Graphical User Interface](#)
- [The Display Preferences Menu](#)
- [Using *BlueHDL* as a VCD Results Viewer](#)
- [Using *BlueHDL* with VerilogXL](#)
- [Using *BlueHDL* with ModelSim Verilog](#)
- [Using *BlueHDL* with Icarus Verilog](#)
- [Using *BlueHDL* with SystemC](#)
- [Other *BlueWave* Features](#)
- [Links to Nextest ATE gear](#)
- [Internal Data Structures Available to User](#)
- [Files and Directory Structure](#)

- [Console Commands](#)
- [Problems, Error Messages and Known Bugs](#)
- [Downloading and Installation](#)
- [Support](#)
- [Hardware Requirements](#)
- [Index](#)

Introduction to VHDL, Verilog and SystemC Simulation with *BlueHDL*

Welcome to the *BlueHDL* User's Manual. It explains the features of the *BlueHDL* simulation tools and provides several short tutorials. These tools provide behavioral and RTL HDL simulation running under Unix, Linux, and Microsoft Windows. For those new to VLSI design, you may need the above acronyms defined: HDL means hardware description language, the most common HDL's are VHDL and Verilog. RTL means register transfer level, a particular coding style required by synthesis tools in order to turn your HDL code into gates and transistors. The tools are intended for both professional and educational use. The Pro 1 Version provides professional quality at the best possible price/performance point and the Student Version is free.

The *BlueHDL* tools consist of: HDL compilers, a simulation engine and a graphical user interface (GUI). A VHDL compiler is available now, and a Verilog compiler will be available in the future. *BlueSim* is the HDL simulation engine. *BlueWave* is the GUI which allows the user to view simulation results of *BlueSim* and other simulators such as Cadence* VerilogXL, Mentor ModelSim, Icarus Verilog and Open SystemC. The GUI formats simulation results into both graphical and tabular formats, allowing the user to develop and debug VHDL, Verilog and SystemC RTL and behavioral mixed-signal designs.

High-Speed Waveform Display

The main job of the *BlueWave* GUI is to display waveforms in the Waves Window to allow designers to debug their designs. A key idea for waveform display tools is "the faster, the better", because the faster the tools responds to your commands, the faster you can debug your designs. *BlueWave* is very file-oriented, which does limit its performance at times. However, *BlueWave* also has some special features which speed up waveform drawing, and when you use these features you should be happy with the performance. The Student Version is quite fast with or without enabling the special features due to its limits on the number of signals and events. However, the Pro 1 Version has significantly higher limits and can be slow when viewing large data sets if not set up correctly.

There are three main special features that allow the Pro 1 Version to display waveforms efficiently for large simulation data sets:

- Local Zoom Redraw
- Rectangle Draw
- Disabling output file creation

All of these features are controlled by the [Display Preferences Menu](#). You use the 'Edit -> Edit Display Preferences' Menu command in either the Navigator or Waves Windows to access this menu.

Local Zoom Redraw means that the Zoom Box, Zoom In/Out and Zoom C0-C1 functions all cause data to be drawn in the Waves Window for only the portion you want to magnify. If you want to be able to scroll through the data on both sides of the zoom area, you can use the Expand function after zooming in. Setting the 'expandwav' Tcl variable to 'no' supports this approach. Setting 'expandwav' to 'yes' means that the tool will always expand after a zoom in, causing degraded performance.

Rectangle Draw means that the tool draws rectangles instead of complex waveforms if more than a certain number of transitions would be displayed in the Waves Window. This feature is enabled by setting the 'rectwav' Tcl variable to 'yes', and the threshold which determines when this feature takes effect is the 'rectthreshwav' Tcl variable. This feature greatly speeds up drawing when you have signals such as clocks with many transitions.

Disabling output file creation speeds up the initial reading of the simulation results file and waveform drawing. This does not affect the drawing speed once simulation data has been read in. *BlueWave* currently creates two types of output files: List files and Ate files for Nextest ATE testers. List files are tabular text files showing simulation results, and Ate files are tabular text files for use with VLSI Automatic Test Equipment (ATE). It is not necessary to create these files very often, and if output file creation is disabled, the speed of reading in results is greatly improved. The 'filemaskwav' Tcl variable controls output file

creation. For the best performance set this variable to 'waves'.

If you are using the Pro 1 Version, you will probably want to set the 'expandwav, rectwav and filemaskwav' Tcl variables as suggested above. By alternating between zooming in and zooming out to fit with occasional expands, you should be able to see all the waveform data needed to debug your designs in a fast, efficient manner.

BlueHDL Tutorials

The *BlueHDL* tools were designed to be easy-to-use. However, all Electronic Design Automation (EDA) tools have an inherent complex nature, and the *BlueHDL* tools are no exception. This user's manual will lead you as far down into the proverbial iceberg as you want to go. In order to start using *BlueHDL* and be productive, you only need to explore the tip. As a designer you may only want to go that far. The GUI provides you with point-and-click means to customize window size and placement, waveform colors, and signal/variable properties. Below this level, you have to begin changing the Tcl/Tk code to more fully customize the GUI.

If you are a programmer, you will probably want to check out a bigger chunk. If you are a masochist, you can dig down into the more than 7000 lines of Tcl/Tk code which comprise the GUI.

We think the best way to introduce you to the tools is to lead you through three simple tutorials. Then you can read the rest of the User's Manual (this html file, blueuser.html). But some of you will be more comfortable reading the rest of this User's Manual before doing the tutorials. Of course, both approaches will get you to the same place: surfing the blue waves!

In all of the instructions below, left click or left double click means single or double click with the Left Mouse Button; right click or right double click means single or double click with the Right Mouse Button.

Introduction to Tutorial 1

This tutorial takes you on a quick tour of the tools. After you have changed directories to the test directory (explained in detail below), you can compile a design and a test bench, simulate and display results with a total of only three button clicks.

The tutorials give you a choice of either compiling VHDL or using precompiled examples. If you are using *BlueHDL* with VerilogXL, ModelSim, Icarus Verilog or Open SystemC, run through tutorials 1, 2 and 3. Then check out the Verilog or SystemC hyperlinks listed above in the [Table of Contents](#).

Start the *BlueHDL* tools under Linux or Solaris

1. In Linux or Solaris enter 'bws' for the free Student Version or 'bwp1' for the Pro 1 Version in an X window, depending on which version you're running. Your PATH environment variable must include the correct path and your BLUEPC environment variable must point to your install area. You also must ensure that the shared libraries required by *BlueHDL* are available.

This may involve setting the LD_LIBRARY_PATH environment variable and/or adding symbolic links to existing shared libraries. You should cd to \$BLUEPC/bin and run ldd on all the files in this directory, for example: ldd bwstud. If you see any messages saying a library was not found, you need to find the closest library you have installed in /usr/lib, /usr/local/lib, /usr/openwin/lib, etc. and create a symbolic link (probably as root). For example, if ldd bwstud produced "libX11.so.6 not found" and you have libX11.so.4 installed, you need to cd to your X11 library area (probably /usr/openwin/lib or /usr/X11R6/lib) and create a symbolic link like so:

```
In -s libX11.so.4 libX11.so.6
```

You will also need to ensure that Tcl/Tk Version 8.3 is installed and visible to *BlueHDL*. You may also need to create some symbolic links for Tcl/Tk. Under Linux and Solaris the easiest way to test Tcl/Tk is to enter 'which wish8.3' (or whatever Tcl/Tk release you have). Your system should tell you where the Tk shell wish8.x lives. Then you should be able to enter: 'wish8.3' and see a small Tk screen pop up.

Start the *BlueHDL* tools under Windows

1. In Microsoft Windows either left double click on the *BlueHDL* icon, or enter 'bwstud' or 'bwpro1' in a DOS window, (depending on which version you're running).

Tutorial 1

2. Use the Compile Browser Window to change directories and compile.

Change directories to the \$BLUEPC/myhome/test dir where \$BLUEPC is the installpath environment variable. The tool should have started up with you already in the \$BLUEPC/myhome dir, so just cd to test by left double clicking on 'test' in the Compile Browser Window.

If you are using VHDL, compile the alu.vhd file by left double clicking on the alu.vhd file entry in the VHDL/Verilog/SystemC Source Files Pane (the Middle Pane). If you are using not VHDL, skip to Step 3b.

You should see a 'compilation successful' message in the Console Window. Repeat the process for the tb_alu.vhd file. You should now see two entries in the Compile Window Top Objects Pane. Left double click on the 'tb_alu(arch_tb_alu)' entry in the Top Objects Pane and the Top Object Box should show tb_alu(arch_tb_alu).

NOTE: Another file named 'alumon.vhd' is provided for the purpose of demonstrating the use of Verilog system tasks such as \$monitor, \$display, \$write and \$time within VHDL. You can also look at other files in this directory for more details on how to use these Verilog system tasks in your VHDL code.

3a. Use the Waves Window to run a simulation and view the results in a graphical format.

Deiconify the Waves Window by left double clicking on its icon. Enter the number 200 in the Run Time: Box followed by a return. Click on the Run Simulation Button to run a simulation and display waveforms. You should see a 'BlueSim elaborating...' message in the Console Window. After about 30 seconds you should see waveforms displayed in the Waves Window. The first time you run a simulation, the results take a little extra time to appear, because the simulator must perform some setup operations and elaborate the design. Successive simulation runs should execute more quickly. If you want to run the simulation a bit longer, click on the Run Simulation Button again. Once you've stopped the simulator, new simulation runs will start from time 0 and will take the extra time for setup and elaboration. You can use the Restart Button to start simulations from time 0 without paying the setup and elaboration time penalty, if the simulator is still running.

3b. Use the Navigator and Waves Windows to load simulation results and view them in a graphical format.

If you are not using VHDL, you can load precompiled Verilog results for tb_alu with the Navigator Window. Choose 'File -> Load Sim Results File.vcd'. Double click on 'tb_alu.vcd' to load the results. You should see waveforms displayed in the Waves Window and signals/variables in the Navigator Window. You can now continue with the tutorial as if you had compiled the tb_alu design.

4. Zoom around the Waves Window and check values.

Click on the Waves Window Fit Button to expand the view. Now hold down the Right Mouse Button while moving the mouse to the right to draw a Zoom Box in the Waveform Pane. Left click on the Zoom In Button a couple of times, and then left click once on the Zoom Out Button. Now left click on the Zoom Expand Button to enable scrolling to the left and right of the zoom area. You can also use these keyboard keys to zoom: 'i' to Zoom In, 'o' to Zoom Out and 'f' to Fit.

The Zoom Box, Zoom In/Out and Zoom C0-C1 functions all cause data to be drawn in the Waves Window for only the portion you want to magnify. This means you get lightning fast redraws, even for large simulation data sets, because data you don't care about is not drawn. The Zoom Expand allows you to selectively redraw the entire simulation data set, in order to scroll through the results. If you prefer to always have a scrollable area after zooming in, you can set the 'expandwav' Tcl variable to 'yes' via the Navigator Window 'Edit -> Edit Display Preferences' Menu command.

Left click anywhere in the Waveform Pane to place Cursor C2. The values of the waveforms at time C2 will now be in the Value at C2 Pane. Place the mouse cursor on any waveform and then use the left arrow and right arrow keys to snap the C2 cursor backwards and forwards to the closest transition. You can also snap by using the 'l' and 'r', '<' and '>' keys.

Click on the Find Button to home the cursors, and then use the Left Mouse Button to drag and drop cursors C0 and C1 to different locations. Now left click on the Zoom Controls: C0-C1 Button to zoom into the area defined by these cursors. You can also set the location of the cursors by entering values into the Cursor Controls: C0, C1 and C2 Boxes. The time differences between the cursors appears in the Measure Cursor Delta Boxes at the bottom of the Waves Window.

5. Use the List Window to view the simulation results in a text format.

Deiconify the List Window by left double clicking on its icon. Choose 'File -> Open List File' from the List Window Menubar.

Left double click on the `tb_alu.lis` item and you should see simulation vectors in the List Pane. Enter the bit pattern 101 in the Search For Box and the List Pane should show the pattern highlighted in red.

Introduction to Tutorial 2

This tutorial takes you deeper into the tools. It shows you how to customize window size, placement and colors in the GUI. It also demonstrates how to use the Navigator Window to select and modify signals and variables for display in the Waves waveform Window and List Window.

Tutorial 2 GUI Customization

You can easily customize much of the GUI by using mouse point-and-click commands. For example, you may want to turn off Balloon Help after you are more familiar with *BlueHDL*. You can do this by using the 'Edit -> Edit Display Preferences' command in the Navigator or Waves Windows, and setting the 'balloonhelpwav' Tcl variable to 'no'. Or you may want to change the window size and placement of the main windows. To do this, just use the mouse to drag and drop window corners and the windows themselves where you want. After changing window size and placement or modifying other user preferences, save the results with either the Save Button in the Display Preferences Menu or with the 'Edit -> Save Display Preferences' Menu command in the Navigator or Waves Windows. This will overwrite the `$BLUEPC/myhome/bwpref.tcl` file. You can also customize each signal/variable to have unique colors or bus formats using point-and-click commands. The second part of Tutorial 2 shows you how. See [The Display Preferences Menu](#) section for details on all the Display Preferences.

For more customization you can modify the `bwsetup.tcl` and/or `bwuser.tcl` files. The `bwsetup.tcl` file contains background colors, fonts and data that isn't likely to change once you're set up. The `bwuser.tcl` file contains data that may change and data that must be loaded after the main file `bwwin.tcl` is loaded. The `bwsetup.tcl` file is in `$BLUEPC/usr` and the master copy of `bwuser.tcl` is in `$BLUEPC/myhome`. You can copy this file along with `bwpref.tcl` to a local design area in order to customize each design environment differently. *BlueHDL* will use the local file if there is one. Otherwise, it will use the master copy in `$BLUEPC/myhome`. Note that after modifying these files, you need to restart *BlueHDL*. These files are self documenting, and eventually you'll probably want to skim through them to see what type of customization is available.

If you want to take total control of the user interface, then you can modify `$BLUEPC/usr/bwwin.tcl`. This file is the user interface. If you modify any of the *BlueHDL* Tcl files and accidentally add a syntax error that prevents *BlueHDL* from loading, don't panic. Just copy the backup file from `$BLUEPC/backup`.

The following example shows you how to edit `bwsetup.tcl` to change a color scheme. To change the Waves Window color scheme from white text over a black background to black text over a white background, use your favorite text editor to make these 2 changes in `$BLUEPC/usr/bwsetup.tcl`:

```
set canvasbgcwav white
set textcolorwav black
```

To change the color for the buttons in all the windows to green make these 2 changes in `$BLUEPC/usr/bwsetup.tcl`:

```
set butcolorwav green
set hotbutcolorwav green
```

Now exit and restart the simulator to see the changes. You can exit from any one of the GUI windows by using the 'File -> Exit' command. Check out the `colors.txt` file in `$BLUEPC/usr` for a few examples of colors you can use for list boxes background, canvas background, etc.

Tutorial 2 Signal/Variable Display Overview

After a design has been successfully compiled and simulated once, the Navigator Window is the next point of user interaction with BlueWave. It provides a view of the design hierarchy. It can also be used to create and edit files that control the order and format of displayed simulation signals and variables.

The main point to remember with the Navigator Window is: if the Sigs/Vars Display List Pane is empty, the Display Level is used to automatically to control the Waves Window and List Window displays. If the Display List Pane is not empty, the order and format of the signals and variables in the Display List Pane are used to control the displays. When you load a previously created `file.dis` in the Navigator Window, that data is used to control the displays. In each of the above cases, the actual file

performing the control behind the scenes is `_bwfile.dis`.

When the Sigs/Vars Display List Pane is empty, set the following values in the Display Level Box to display different numbers of signals:

- 0 to display all the signals/variables in the design.
- 1 to display the signals/variables in level 1 only (top level).
- 2 to display the signals/variables in levels 1 and 2, etc.

Remember that you can only display as many signals and variables as your version will allow. See www.bluepc.com/bluehdl.html for the limits.

Tutorial 2 Signal/Variable Display Details

This part of Tutorial 2 shows you different ways to select signals/variables for display, and some other Navigator Window features such as signal/variable property edit, name search and labeling.

1. Load the `tb_alu.vcd` file.

First, make sure that the Compile Browser Window Current Dir displays `$BLUEPC/myhome/test` as the current directory. If not, double click on entries in the Compile Browser Directories Pane to `cd` there. Choose 'File -> Load Sim Results File.vcd' in the Navigator Window. Double click on 'tb_alu.vcd' to load. There should now be entries in all 3 panes of the Navigator Window.

2. Use the Navigator Window to select signals/variables for display.

Click on the Navigator Window Clear Button to clear the Sigs/Vars Display List Pane. Left click on the 'tb_alu module' entry in the Entities/Module Pane, then right click. You should all of the tb_alu signals/variables appear in the Sigs/Vars Display List Pane.

Hold down the Left Mouse Button and drag the Mouse Pointer over the last three entries in the Sigs/Vars Display List Pane to select them. Now left click on the Cut Button to delete them.

Left double click on the 'alu module' entry in the Entities/Module Pane. You should see all of the alu signals/variables appear in the Sig/Var Pane. Left click on first signal, then hold shift and left click on the third signal. Now right click and you should see these 3 signals appear in the Sigs/Vars Display List Pane. You should now have 8 signals in the Sigs/Vars Display List Pane.

3. Display the waveforms.

Left click on the Update Waves Button. You should now see the waveforms for these 8 signals appear in the Waves Window.

4. Use the Navigator Window to change signal/variable properties.

Left double click on the `tb_alu/a` item in the Sigs/Vars Display List Pane. The Sig/Var Edit Window will pop up and the selected signal/variable should now appear in the Sig/Var Name and Format Boxes. Click on the 'split' Radio Button and then click on the Apply Button. You should now see the 4 bit bus split into its 4 individual bits in the Waves Window. You can also toggle the bus display format between split and default format by left clicking on the bus name in the Waves Sig/Var Names Pane.

Also try some of the other choices: hex, binary, decimal, octal and analog. Now left click in the white Color field and the Color Palette should appear. Choose a new color, click OK, then click on the Apply Button.

5. Search for a name.

If you have a large Sigs/Vars Display List, you may not be able to easily find a particular signal/variable name in the Waves waveform Window. You can use the name search feature to locate a name. Enter the string `*s*` in the Search Box followed by a return. You should see the three status lines (`s2`, `s1` and `s0`) marked with colored dots in the Waves waveform window. Click the Off Button to turn off the search markers.

6. Add some labels.

To better organize your signals/variables in the Waves waveform Window you can add some labels. Left click to select an entry in the Sigs/Vars Display List Pane. Then enter a label followed by a return in the Label Box. As with standard programming

labels you can use underscores (`_`), but no spaces.

7. Save and reload display formats.

Once you are happy with the order and format of the signals/variables in Waveform and List Windows, you can save file.dis for future use. Choose 'File -> Save Display File.dis' in the Navigator Window, enter a filename in the File Name Box (without the .dis extension) and click on the Save Button.

You can also load previously saved file.dis files. You may first have to cd to a working area that has a `_bluewav` directory containing the file.dis of interest and a corresponding simulation results file.bcd, then load a file.bcd to initialize the design tree hierarchy information.

After these conditions are satisfied you can load file.dis: choose 'File -> Load Display File.dis' in the Navigator Window, and double click on a filename.

Introduction to Tutorial 3

This tutorial demonstrates how to search the Waves Window for particular signal/variable values. It also introduces you to the User Proto Window, which is a product prototype window.

Tutorial 3

1. Search the waveforms for values.

You can search the Waves Window for particular signal/variable values, including finding the intersection of values for up to three different signals/variables. You can use either the Search Menu or enter search commands in the Console Window. The following examples assume either you have just run a 200 ns simulation of `tb_alu` as in Tutorial 1, or you have used the 'File -> Load Sim Results File.vcd' command to reload `Tb_alu.vcd` file as in step 2a above.

First, bring up the Search Menu by choosing 'Search -> Search Menu' in the Waves Window. Then fill in the Search Menu like this:

```
1st Index Box: 1, 1st Value Box: 2
2nd Index Box: 2, 2nd Value Box: f
3rd Index Box: 3, 3rd Value Box: 1
Leave the Start at Time Value as 0
```

You can fill in the Index Box or just click in the Name Box if a name is already selected in the Navigator Sigs/Vars Display List Pane. Be sure to use a carriage return if you use the Index Box data, this will cause the name to appear in the Sig/Var Name Box. Now click the Search Button. The results should now be displayed in the Console Window with a message saying 'Search target found at ...', and also in the Waves Window with a red Search Cursor 'Found' moving to that location. Delete the Search Cursor with the Delete Button. Clear the values with the Clear Button. Hide the Search Menu with the Hide Button.

You can customize the `sx` and `initsm` procedures in `bwuser.tcl` if you need to search often for a particular set of values. To search for VHDL enumerated type values, use the index of the enumerated type as the value based on the order the enumerated states in your source code. The 1st enumerated state is index 0, the 2nd is index 1 and so on (in contrast to signal/variable indices which always start with 1).

You can also use Console commands to search the Waves Window. To achieve the same result as above, enter this command in the Console Window:

```
s3 1 2 2 f 3 1 (ret)
```

The `s3` command actually calls the more general `searchw` command. The `s3` command always starts with time 0, but the `searchw` command allows you pass in the time to start searching. There are also `s1` and `s2` commands to search for only 1 or 2 values. The more general `searchw` command for the above data is called like this:

```
searchw 0 1 2 2 f 3 1 (ret).
```

Its arguments mean: search the Wave Window starting at time 0, and search signal/variable 1 (the index listed in the Sigs/Vars Display List Pane) for the value 2, signal/variable 2 for the value f, and signal/variable 3 for the value 1. The `searchw` command takes 7 arguments: a time, and 3 signal/variable index and value pairs. To delete the Search Cursor using a Console command

enter: dels (ret). You can also use the sets t command (set Search Cursor at time t) to place as many markers as you want, like this: sets 100 (ret).

2. Activate the User Proto Window.

The User Proto Window provides you with a means to create a product prototype window. You can customize this window to look like the interface to whatever product you are developing. For instance, if you are developing HDL code for a cellular phone, you could program this window to look like the phone keypad and bit mapped display. You can use this window to view the simulation results in any type of format and to create input stimulus for your design.

Deiconify the User Proto Window. The listbox shows the final values for the signals and variables whose indices are listed in the dislistwav list in bwuser.tcl. If you enter data in the Input Data Box or press one of the buttons, a file named myinput.txt is written to the current directory.

The code in bwproto.tcl, which defines this window, is meant to be only a starting point. You can use this window to view the simulation results in any type of format and to create test bench input files. Here you are only limited by your imagination and your Tcl/Tk programming skills.

BlueHDL Help

BlueHDL provides four levels of help. Within *BlueWave* you can enter either '?' or 'help' in the Console Window to see the Console commands. You can select items from the Help Menu in any of the windows. Balloon Help automatically appears if the mouse cursor sits for a few seconds over many of the fields in the windows. You can access the *BlueHDL* User's Manual (this html file, blueuser.html) from either the distribution files at \$BLUEPC/html, or from our Web site at www.bluepc.com to get the latest version.

Balloon Help is very useful for a new user in the first few sessions. But after that, it becomes more of an annoyance. You can turn off Balloon Help by setting the 'balloonhelpwav' Tcl variable to 'no' via the Navigator Window 'Edit -> Edit Display Preferences' Menu command.

A Short History of the *BlueHDL* Tools

The three main themes of the *BlueHDL* tools are:

- Professional quality HDL tools don't have to cost a bundle.
- HDL simulation tools should help make design fun and easy.
- VHDL and Verilog should be unified and run under multiple OS's.

We began teaching VHDL at the University of California in San Diego in 1995 and were immediately faced with the problem of providing our students with a simulator that they could use at home. We weren't able to find one. We found a couple of low-cost or free demo simulators that were somewhat usable, but more trouble than they were worth. We convinced UCSD to buy Modeltech (now ModelSim) at a generous academic discount. But the students could only use this simulator in the campus computer laboratory.

This experience led us to our first main theme: professional quality tools for students and individual engineers. We began to think that if we could create an HDL simulator that had professional quality features, but that students and individual engineers could afford, we might have a viable product idea. Our first business model was to give away for free a very limited Demo Version, charge \$49 or \$99 for a Student Version, and charge \$495 or \$995 for a Professional Version. After some more thought we decided to dump the Demo Version, give away the Student Version for free and only charge for the Professional Version. In 2000 we added support for SystemC and in 2001 we increased the limits for the Student Version.

Easy to Use

The second main theme listed above that HDL design can be fun and easy led us to:

- Keep the number of windows to a minimum.
- Make all main functions visible (rather than hiding under menus).
- Minimize setup required for compilation, simulation and results display.

A consequence of this approach is "one-button compile" and "one-button simulation". That is, both the HDL compilation and

simulation processes can often be performed with only the touch of a single button. This allows the designer to concentrate on the design instead of the tool, and the result is that the HDL design experience is actually fun!

The "one-button compile" feature means you don't have to set up libraries ahead of time. The work library is created automatically. Just use the Compiler Browser Window to change directories to your source file location and double click on a VHDL/Verilog/SystemC source file to compile.

The "one-button simulation" feature means you don't have to click on a lot of buttons and menus to see waveforms. Once a design and test bench have been successfully compiled, simply pushing the Run Simulation Button will run a simulation and display simulation waveforms. All the test bench signals/variables simulation results will be displayed automatically in graphical waveform and tabular list formats. Of course, you can also navigate the hierarchy and selectively choose signals/variables from any level of the design hierarchy for display as shown in Tutorial 2.

Multiple HDL's under Multiple Operating Systems

We have been strongly committed from the start to provide support for both VHDL and Verilog in a unified simulation environment. We also believe that the C/C++ HDL's have a promising future. The Open SystemC movement has convinced us to add the SystemC HDL as a third language. While we don't see C/C++ HDL's replacing VHDL and Verilog, they provide a better way to perform hardware/software co-simulation.

We chose our programming languages and development environment to enable us to offer our tools under the three main operating systems, Linux, Unix and Windows. The choice of the GNU tools and Tcl/Tk has made our porting efforts relatively painless, and we now offer our tools under all three operating systems.

Tcl/Tk as the GUI interface

The *BlueWave* GUI is written in C and Tcl/Tk. The use of Tcl/Tk for a graphical user interface is a growing trend among EDA tools, from simulators to timing analysis tools. Tcl/Tk provides a very customizable, flexible, open source interface. The user can utilize a combination of a command shell with history and the more standard GUI buttons and menus with a mouse. The beginning user will tend to favor the buttons, menus and mouse, while the more experienced user will often find the command line interface more efficient.

Four Different Levels of Customization

We have used Tcl/Tk to provide four different levels of customization. The first level is via point-and-click mouse controls. You can interactively change default colors and other key user preferences, and you can also resize, position and iconify windows. You can then save these preferences to be used during the current session and in future sessions. To change default colors and other user preferences use the 'Edit -> Edit Display Preferences' Menu command in the Navigator or Waves Windows. To change windows just size and place the windows where you want, or iconify them. Then use the 'Edit -> Save Display Preferences' Menu command. This will overwrite the \$BLUEPC/myhome/bwpref.tcl file where user preference information is stored.

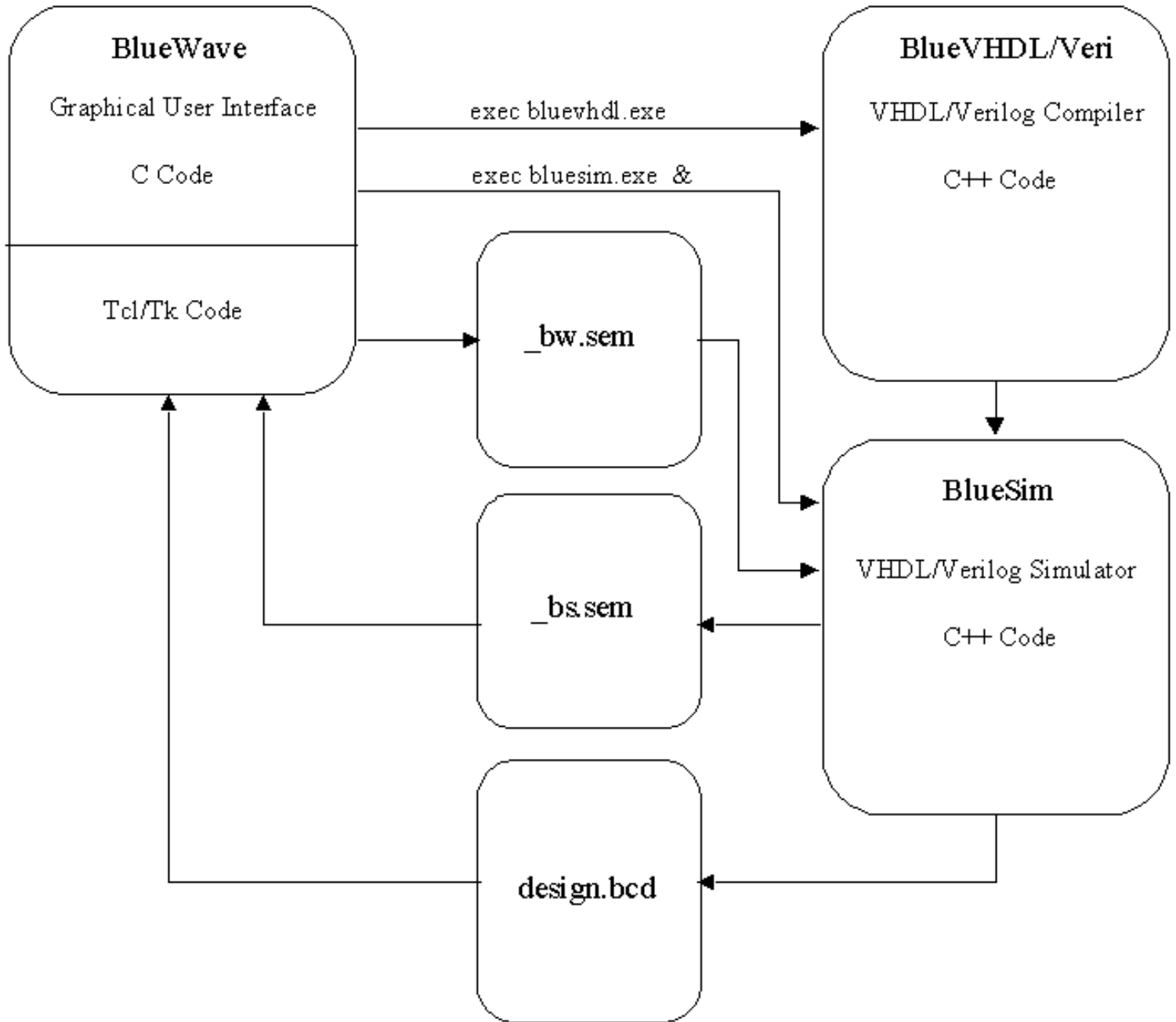
The second level is with the bwsetup.tcl and bwuser.tcl files. They provide additional Tcl/Tk variables that allow you to easily control such aspects as: text fonts, logical to physical mappings and so on. The bwsetup.tcl file contains data that is not likely to change once you're set up, and bwuser.tcl contains data that is likely to change or data that must be set after the main Tcl file bwwin.tcl is loaded.

The third level is with the bwproto.tcl file, which provides a prototype window. You may not want to use the Proto Window, but its intention is to allow you to customize a window to look like the interface to whatever product you are developing. Take a look at this file, which is in the \$BLUEPC/usr directory. It's not that big (a little over 100 lines of code), and it's pretty easy to see how the code maps to the User Proto Window that it creates. If you're not already a Tcl/Tk programmer, this file could change your life! We're only partly kidding. It actually could be your passport to the fun world of Tcl/Tk programming. See [below](#) for more details.

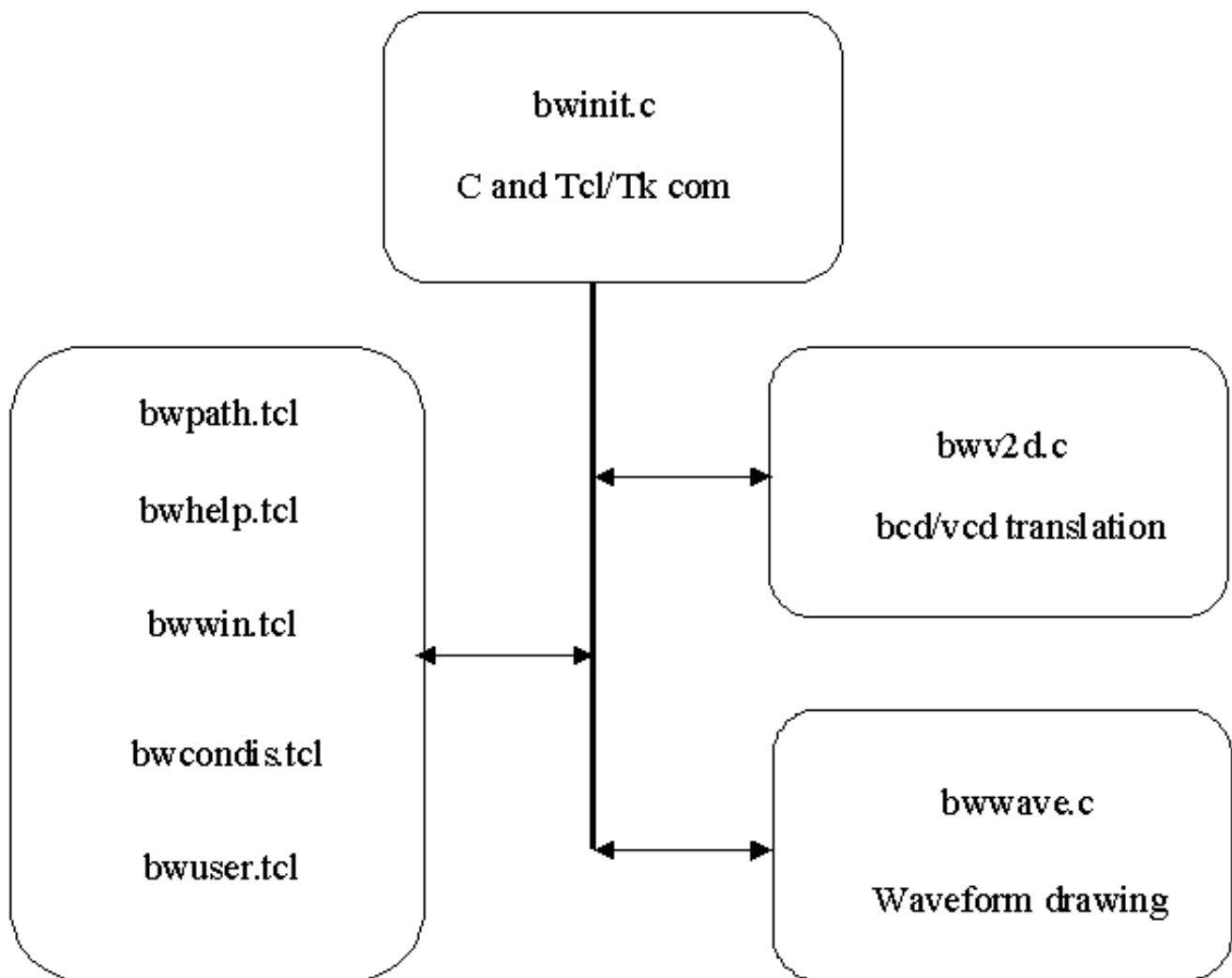
The fourth level is with the bwwin.tcl file, which provides complete access to all of the window code. This allows the experienced Tcl/Tk programmer to fully customize the interface. It also creates an opportunity for a student to learn Tcl/Tk programming. For more details about Tcl/Tk see: [More About Tcl/Tk](#).

The following two figures show the basic software architecture.

Blue HDL Architecture 1



Blue HDL Architecture 2: Structure of BlueWave GUI



BlueHDL Details

The following sections provide details on the separate programs that make up the *BlueHDL* tools: the language compilers, the *BlueSim* simulator and the *BlueWave GUI*.

The HDL Compilers

BlueHDL currently features a VHDL compiler. A Verilog compiler will be available in the future. The *BlueWave GUI* supports any HDL simulator that can output a VCD file, including other Verilog simulators and Open SystemC.

The HDL compilers feature simulation/synthesis syntax checking and support the usage of Verilog system tasks and functions for debugging purposes. Simulation syntax checking finds basic language syntax errors. Synthesis syntax checking looks for multiple drivers and incomplete sensitivity (event) lists. The Verilog system tasks can be embedded in either VHDL or Verilog source files. For VHDL compilation, users can embed these system tasks in the VHDL code and compile using the default setting of BlueVHDL. To turn this support off, simply change the setting of the 'vlogsyswav' Tcl variable in the bwsetup.tcl file to "no" for the GUI, or set the option "vlog" option to "no" for a batch compilation.

The supported Verilog system tasks are:

- \$display, \$displayb, \$displayh
- \$write, \$writeb, \$writeh
- \$monitor, \$monitorb, \$monitorh
- \$time

All the control and format specifications specified in the Verilog IEEE-1364-1995 LRM are supported for decimal, binary and hexadecimal display, write and monitor tasks. The \$monitor tasks can only be called once in each process and they support all control specifications for VHDL signals. The display and write tasks support all control specifications for all VHDL objects (including signals, variables,...,etc.). See files: "vlog.vhd", "almon.vhd" and "slmon.vhd" in the directory "\$BLUEPC/myhome/test" for more examples.

The HDL compilers can be operated from a command shell in batch mode. To obtain the syntax for the VHDL compiler, simply type:

bvhdx (ret) where x is either stud or pro1.

Then you should see something like:

usage: bvhdstud [-lib {libname}] [-syn] {sim/synth} [-vlog] {y/n} {filename}

options: -lib: logical library name (default: work)

-syn: syntax check (default: from bwpref.tcl)

-vlog: verilog system task support (default: from bwsetup.tcl)

example: bvhdstud test.vhd

For example in the Student Version, to compile the file alu.vhd into library mylib, enter:

bvhdstud -lib mylib alu (ret)

BlueSim

The simulation engine is language independent and is capable of simultaneously simulating modules written in VHDL, Verilog and C++. The engine itself is written in object oriented C++, and it takes full advantage of modern C++ optimization. It will allow future expansion to to a multithreaded and cycle-based engine. It also features the Verilog printf-like \$monitor, \$display and \$write debugging commands for both VHDL and Verilog.

The simulator can be operated from a command shell in batch mode. To obtain the syntax for the simulator, simply type:

bsx (ret) where x is either stud or pro1.

Then you should see something like:

```
usage: bsstud [-lib {libname}] [-sh {(a/o)}] [-t[imeunit] {fs/ps/ns/us/ms/sec/min/hr}]
[-r[untime] {simtime[{unit}]}] [-ch] {top_design_name}
```

options: -lib: logical library name (default: work)
-sh: BCD output mode (append/overwrite) (default: append)
-t[imeunit]: sim. resolution (default: from bwpref.tcl)
-r[untime]: simulation run time (default: 100ns)
-ch: compile hierarchy (no sim) (default: off)
example to simulate topdesign for 5 milliseconds: bsstud -r 5ms topdesign

For example in the Student Version, to simulate the top level object tb_alu from library mylib for the default of 100 nanoseconds, enter:

```
bsstud -lib mylib tb_alu (ret)
```

BlueWave

The *BlueWave* tool is a graphical user interface (GUI) which allows the user to control simulation and to display simulation results in a graphical waveform display format and in tabular formats. It can display and format results from the *BlueSim* simulator and foreign HDL simulation engines that provide Verilog value change dump files (VCD). Foreign simulators supported include Cadence* VerilogXL, Mentor ModelSim VHDL and Verilog, Icarus Verilog and Open SystemC. *BlueWave* consists of six main windows:

- [Console Window](#)
- [Compile Browser Window](#)
- [Navigator Window](#)
- [Waves Waveform Window](#)
- [List File Window](#)
- [User Proto Window](#)

All compilation, simulation and results viewing takes place within these windows.

Console Window

The Console Window allows the user to enter short commands to control the simulator and to read warning and error messages from the tools. Because the interface is written primarily in Tcl/Tk the user can easily extend the commands available in the Console Window. Enter '?' or 'help' in the Console Window to see a list of the commands.

In all of the instructions below left click or left double click means single or double click with the Left Mouse Button; right click or right double click means single or double click with the Right Mouse Button.

Compile Browser Window

The Compile Browser Window is the first point of entry for a user. It allows the user to change directories to his/her working area and to control the compilation process. You may not even need to change directories because *BlueWave* remembers the directory you were using the last time you exited the tool, and it automatically sets up the Compile Browser Window in this

directory on start up.

Left double click on an directory item in the Directories Pane to change directories. The Compile Browser Window is not only for compiling, it is also a general directory browser. Use it to change directories to access previously compiled or simulated data.

Next, left double click on any VHDL, Verilog or SystemC source file in the VHDL/Verilog/SystemC Source Files Pane (the Middle Pane) to compile. Look in the Console Window for error messages and fix any syntax errors. Re compile until you have clean results for both a design and test bench.

Then left double click on any top level object in the Top Object Pane to select a top level object for simulation. Top level objects are usually either entity names or module names in the test bench.

From here you can move directly to the Waves waveform Window and simulate. Simply left click on the Run Simulation Button in the Waves waveform Window and you should see waveforms. Or you can more closely control what you see by using the Navigator Window, which is described next.

Navigator Window

The Navigator Window provides the user with a means to traverse the design hierarchy and to select and modify signals and variables for display. It is responsible for the display order and format of signals and variables for all windows and files: the Waves waveform Window, the List File Window, Waveform PostScript files, list files, spice files and test vector files. It features editing functions like add, cut, copy, paste, clear, drag and drop.

BlueWave differs from some other HDL simulation tools in providing this one central point for managing the display of signals and variables, instead of multiple points where each window separately manages the display of signals and variables. Our approach makes for an easier-to-use tool because multiple windows and files are organized at the same time. It also provides for faster response because the list-based Navigator can efficiently manage the lists of signals and variables while the graphics-based Waves waveform Window and text-based List Window and associated files can efficiently process their specialized graphical and textual data.

Navigator Window Details

If the Sigs/Vars Display List Pane is empty, *BlueWave* uses the Display Level Box to control the display of signals and variables. Set the following values in the Display Level Box to display different numbers of signals:

- 0 will display all the signals/variables in the design.
- 1 will display the signals/variables in level 1 only (top level).
- 2 will display the signals/variables in levels 1 and 2, etc.

But remember that all the versions have some limits on the number of signals/variables that can be displayed. If the Display Level value would cause that number to be exceeded, some signals/variables won't be displayed.

The following explanation of the Navigator Window refers to three window panes: the Entities/Modules Pane (upper left pane), the Sigs/Vars Pane (lower left pane), and the Sigs/Vars Display List Pane (right pane). It's helpful to have the Navigator Window displayed in front of you while reading the following.

If the Sigs/Vars Display List Pane is not empty, then *BlueWave* will display the signals/variables listed in the Pane. The easiest way to place signals/variables in the Sigs/Vars Display List Pane is to select an Entity/Module in the Entities/Modules Pane with a left click, then right single click to add. You can also select individual signals/variables in the Sigs/Vars Pane with a left click to select and a right click to add the selected items to the Sigs/Vars Display List Pane. You can also left click on the Add Button once items have been selected.

The Edit Controls Buttons are used to manipulate the Sig/Var Display List Pane once it contains data. Use the Copy, Cut, Paste and Clear buttons to manipulate the contents and order of this pane.

Navigator Window Edit Properties, Search Name, Labels

The Navigator Window provides the means to edit signal/variable colors and bus formats. It also provides for signal/variable name search, labeling signal/variable groups and vector file utilities. See [Tutorial 2](#) for examples of property edit, name search

and group labeling.

To edit the color of any signal/variable or format of a bus, left double click on any signal/variable in the Sigs/Vars Display List Pane. This will bring up the Sig/Var Edit window. To change a color, left click in the color field (the white entry box). This will bring up the Color Palette. Choose a color, then click OK. To change a bus format click on a Radio Button, then click on the Apply Button. The choices are: hex, binary, decimal, octal split and analog. Split means that the individual bits which make up the bus will be shown separately. Analog means that vectors will also be shown as an analog continuous wave in addition to a digital square wave. Left click on the Apply Button to see the results in the Waves waveform Window.

The name search feature allows you to locate a name in a large Sigs/Vars Display List. Group labeling assists you in better organizing your signals/variables in the Waves waveform Window.

The Navigator Window also has a Utilities Menu which provides translation of Mentor ModelSim VCD vectors to more compact BCD vectors. The Utilities Menu also can be used to pop up a Postprocess Window that allows you to look at subsets of large VCD/BCD files.

Waves Waveform Window

The Waves Waveform Window provides the user with a means to control the simulation and view results graphically. You can start and stop simulations, zoom in and out, measure times and display specific results.

Use the Simulations Controls Buttons to control simulations. The Run Simulation Button runs a simulation for the time specified in the Run Time Box after a design and test bench have been compiled. The Restart Button runs a simulation from time 0. The Stop Button stops the simulator and forces elaboration.

The easiest way to zoom in is to use the Right Mouse Button to draw a Zoom Box around an area. Use the Zoom Controls to zoom in, out, fit and to zoom to a region between the C0 and C1 cursors. Also use these Console Window commands to zoom in, out and fit: zi, i, zo, o, zf, f. After you're happy with the zoom area, click on the Zoom Expand Button to redraw the expanded view, which enables scrolling to the left and right of the zoom area with the horizontal scrollbar.

The cursors allow you to measure times and to set zoom limits. You can drag and drop the cursors with the Left Mouse Button, or you can enter a time value in the C0, C1 or C2 Boxes. Use the Find Button to home the cursors. Cursor Delta values are displayed at the bottom of the Waves waveform Window.

Cursor C2 is used to display results at a specific time. Left click at a time to place the C2 Cursor, and the values for all signals/variables at that time will be displayed in the Value at C2 Pane.

The Sig/Var Pane is not editable. Use the Navigator Window to control the order and format for signals/variables to be displayed.

List File Window

The List File Window provides the user with a means to view the simulation results in a tabular vector format. If the 'deltamodewav' Tcl variable is set to 'yes', delta cycles are displayed next to the time value. If the file.bcd results file contains delta cycle data, then 'deltamodewav' must be set to 'yes'. Use this window's Search For Box to search for specific patterns.

User Proto Window

The User Prototype Window provides the user with a means to create a product prototype window. You can customize this window to look like the interface to whatever product you are developing. For instance, if you are developing HDL code for a cellular phone, you could program this window to look like the phone keypad and bit mapped display. You can use this window to view the simulation results in any type of format and to create test bench input files. Here you are only limited by your imagination and your Tcl/Tk programming skills.

The initial Tcl code behind this window provides access to the final values of any signal/variable via a listbox in bwproto.tcl and the dislistwav list in the bwuser.tcl file. It also provides a data entry box and some buttons that create a one line file which could be read by your test bench.

Some users will leave this window as is, but the intent is to provide you with an easy way to create a prototyping window, and we hope many users will make this window their most important window.

The Display Preferences Menu

The Display Preferences Menu allows you to easily customize most of the *BlueWave* GUI by using mouse point-and-click commands. You can access this menu from either the Navigator or Waves Windows with the 'Edit -> Edit Display Preferences' Menu command. You can save new preferences with either the Save Button in the Menu or with the Navigator or Waves 'Edit -> Save Display Preferences' command. If you wish to temporarily alter a display feature, simply change a value and the new value will take affect with the next Waves redraw. Saving the preferences saves them to \$BLUEPC/myhome/bwpref.tcl for use with future sessions. NOTE: saving the display preferences also saves your current Window locations, so your windows will be placed in the saved positions when you start *BlueHDL* the next time.

The following is an explanation of each field and Tcl/Tk variable in the Display Preferences Menu.

Edit Label Color:

To change label color, single click in a Color Value Field. Choose a color from the Color Palette and click OK.

Edit Default Type Color for Single Bit, Bus, Real (Analog) and Enumerated/Boolean:

This field sets default display color data in file.dis. To change color, single click in a Color Value Field. Choose a color from the Color Palette and click OK. To set display color for an individual signal, use the Navigator 'Edit -> Edit Sig/Var Properties' command.

Edit Default X Value (Unknown) Color:

To change X value color, single click in a Color Value Field. Choose a color from the Color Palette and click OK.

Edit Default Z Value (Tri-state) Color:

To change Z value color, single click in a Color Value Field. Choose a color from the Color Palette and click OK.

Use X Color:

Enable/disable use of special color for unknown values.

Use Z Color:

Enable/disable use of special color for tri-state values.

X Forgive:

Allow this many transitions before flagging X value.

See Text:

Show bus value text in waveform if the value lasts for this many pixels or longer.

Y Space:

Vertical spacing of waveforms in Waves Window.

Run Time:

Simulation Run Time in current Time Units.

Display Level

controls the number of signals to be placed in the Display File (file.dis) based on the hierarchy.

dislevelwav = 0 displays signals in entire design.

dislevelwav = 1 displays signals in test bench only.

dislevelwav = 2 displays signals in first 2 levels.

Event Max:

Memory management control:

eventmaxwav = 0, BlueHDL automatically allocates memory based on event count.

eventmaxwav > 0, BlueHDL allocates eventmaxwav words of memory.

Balloonhelp:

Enable/disable balloonhelp.

Balloonhelp Delay:

Wait this many seconds before displaying balloonhelp.

Rectangle Draw:

Enable/disable drawing rectangles instead of complex waveforms in Waves Window when using high zoom factor. Enabling this feature greatly speeds up drawing of large result files.

Rectangle Draw Threshold:

If rectangle draw is enabled, draw a rectangle instead of a complex waveform if more than this number of transitions would be displayed.

Verbose Mode:

verbosewav = 0, only error and warning messages are displayed.

verbosewav > 0, many internal debug messages are displayed.

Search Increment:

Advance next waveform value search by this many time units.

Breakpoints:

Enable/disable use of breakpoints. If enabled, then BlueWave uses breakpoint table to control simulation.

Postscript Mode:

Enable/disable Postscript formatting in Waves Window Waveforms Pane. When enabled BlueWave displays signal name in the Waveforms Pane of Waves Window for Postscript files, since Tcl/Tk Postscript function can only print one pane.

Cursor 2 Search:

Enable/disable use of Cursor 2 to display signal value in C2 Values Pane of Waves Window. Disabling this function speeds up drawing of large result files.

Expand Wave:

Enable/disable auto expansion of waveform view in Waves Window. Disabling this function speeds up drawing of large result files. It's usually better to disable this function and then click on the Expand Button when you wish to use the horizontal scrollbar to scroll through waveforms.

File Mask:

Filemaskwav is the master control for waveform display and output file creation. Choices are:

waves (0x01)

waves&list (0x03)

waves&list&test (0x23)

Display Mask:

Dismaskwav sets default waveform display and output file creation data in file.dis. Choices are:

waves (0x01)

waves&list (0x03)

waves&list&test (0x23)

Bus Display Format:

Busdisformat sets default display format for buses in file.dis. To set a display format for an individual bus, use the Navigator Window 'Edit -> Edit Sig/Var Properties' command. Choices are: hex, binary, decimal, octal, split and analog.

Time Unit:

Choose a time unit for use with the Waves Window. Choices are: microseconds, nanoseconds, sub nanoseconds and picoseconds.

Syntax Check:

Choose a type of syntax checking for the HDL compiler. Choices are: simulation or synthesis. Simulation syntax checking finds basic language syntax errors. Synthesis syntax checking looks for multiple drivers and incomplete sensitivity (event) lists.

The *BlueWave* GUI program within *BlueHDL* can be used as

Using *BlueHDL* as a VCD Results Viewer

The *BlueWave* GUI program within *BlueHDL* can be used as a standalone VCD results viewer. It has several features that make it an easy-to-use, powerful debugging aid such as: waveform value search, signal name search, and signal group labeling.

BlueWave can be used to control foreign simulators, or it can be used simply to view hierarchy, waveforms and tabular list file

results by translating VCD files. If you simply want to view waveform results, you can set up `bwuser.tcl` to withdraw all but the Navigator and Waves waveform windows. If you want to use *BlueWave* to control a foreign simulator, and view results, continue reading the following sections.

Using *BlueHDL* with Foreign Verilog Simulators

These are instructions on how to use Blue Pacific's *BlueHDL* with a foreign Verilog simulator such as VerilogXL, ModelSim Verilog or Icarus Verilog. VerilogXL and Icarus Verilog are well suited for use with either the GUI buttons or the Console Window commands. ModelSim Verilog is best used with Console Window commands.

Using *BlueHDL* with VerilogXL

1. Create a subdirectory called `'_bluewav'` in your Verilog source code directory. Create a file called `'allfiles'` in your Verilog source code directory and include all of your design and testbench filenames in it. For example `allfiles` might contain:

```
tb_alu.v alu.v
```

Set up the creation of a VCD file by placing VCD file generation statements in your testbench file. For example:

```
$dumpfile("_bluewav/verxl.vcd");  
$dumpvars;
```

2. Set up and start *BlueHDL*.

Make sure that `bwuser.tcl` contains the correct settings for foreign Verilog: the `'comptypewav'` and `'simtypewav'` Tcl variables should be set to `'fverilog'` and the `languagewav` should be `'v'`.

The `bwuser.tcl` file is also a good place to set the `'vcdfilename'` Tcl variable. This value needs to match the name you used in the `$dumpfile` command above. For example, if the `$dumpfile` command specifies `'verxl.vcd'`, you should set the `'vcdfilename'` Tcl variable to `'verxl'`. Or you can set the `'vcdfilename'` Tcl variable interactively in the Console Window. For the above example, you would enter: `set vcdfilename verxl (ret)`.

You can use one VCD filename such as `'verxl'` for all of your designs, if you don't want to keep changing this variable. Or of course you can use different VCD filenames for each of your designs, if you prefer.

Start *BlueHDL* with `bws` for the Student Version (or `bwp1` for the Pro 1 Version).

Left click in the Left Pane of the Compile Browser Window to change directories until you are in the directory containing your source code. If you followed the directions in Step 1 you should see a `_bluewav` directory in the Left Pane, and your `files.v` in the Middle Pane. The Right Pane isn't used with foreign Verilog.

3. Compile your design using the GUI buttons in the Compile Browser Window. Either click on the Compile Button or double click on any source file.v in the Middle Pane to compile (this runs the `'verilog -f allfiles'` command).

4. Simulate your design using the GUI buttons in the Waves Window.

Click on the Run Simulation Button. You should see waveforms appear in the Waves Window and signal names appear in the Navigator Window. There is no need to set up display signals first, *BlueHDL* automatically displays the test bench signals in the first simulation run. If you don't place any `'$stop;'` statements in your code, you can even skip using the Compile Window (except to change directories) and run everything from the Waves Window.

Use the Waves Window as described in [Tutorial 1 Zoom Functions](#) to examine the results by zooming and moving the line cursors. The simplest and most useful zoom commands are: Zoom in by drawing a Zoom Box with the Right Mouse Button. Zoom out by clicking on the Fit Button. Note: the Restart and Stop Simulation Buttons are not used with VerilogXL.

Using *BlueHDL* with ModelSim Verilog

1. Create a subdirectory called `'_bluewav'` in your Verilog source code directory. Create a file called `'allfiles'` in your Verilog source code directory and include all of your design and testbench filenames in it. For example `allfiles` might contain:

```
tb_alu.v alu.v
```

Set up the creation of a VCD file by placing VCD commands and the run command in a do file. An example do file named mymti.do:

```
vcd file _bluewav/myvcd.vcd
vcd add *
run 20 us
vcd flush
```

2. Set up and start *BlueHDL*.

Make sure that bwuser.tcl contains the correct settings for foreign Verilog: the 'comptypewav' and 'simtypewav' Tcl variables should be set to 'verilog' and the languagewav should be 'v'.

Start *BlueHDL* with bws for the Student Version (or bwp1 for the Pro 1 Version).

3. Compile your design. In the Console Window enter: vlog -f allfiles (ret).

4. Simulate you design. In the Console Window enter: vsim work.tb_design < mymti.do (ret).

This assumes that you are using the work library (created with 'vlib work'). It also assumes that your top level object is called tb_design and that mymti.do contains the four lines as shown above.

Then use 'File -> Load Sim Results File.vcd' in the Navigator Window to see results in the Waves Window.

Use the Waves Window as described in [Tutorial 1 Zoom Functions](#) to examine the results by zooming and moving the line cursors. The simplest and most useful zoom commands are: Zoom in by drawing a Zoom Box with the Right Mouse Button. Zoom out by clicking on the Fit Button. Note: the Restart and Stop Simulation Buttons are not used with ModelSim Verilog.

You can use the 'Utilities -> Translate Mti Vcd to Bcd' in the Navigator Window to create a BCD file that is more compact the the ModelSim VCD file. Then use 'File -> Load Sim Results File.bcd' in the Navigator Window to load this file and see results in the Waves Window.

Using Tcl Scripts and Procedures to Automate Running ModelSim Verilog

You can also create custom Tcl scripts and procedures that incorporate all of the above functions. An example script named myscript.tcl:

```
#compile
exec vlog -f allfiles
# run the simulator with the mymti.do file that is shown above
exec vsim work.tb_design < mymti.do
# compress the ModelSim vcd vectors into bcd vectors
translatemti /myhome/test/_bluewav/tx_testall.vcd
# load the BlueHDL bcd file and see the results in the Waves Window
loadbcd /myhome/test/_bluewav/tx_testall.bcd
# cd to the source area
cd /myhome/test
```

To execute the above script in the Console Window enter: source myscript.tcl (ret).

Alternatively, you can create a procedure in the bwuser.tcl file that contained that above lines bracketed with the Tcl procedure call syntax like so:

```
proc myscript {} {
contents of myscript.tcl placed here
}
```

To execute the above procedure in the Console Window enter: myscript (ret).

Using *BlueHDL* with Icarus Verilog

1. Create a subdirectory called '_bluewav' in your Verilog source code directory. Create a file called 'allfiles' in your Verilog

source code directory and include all of your design and testbench filenames in it. For example allfiles might contain:

```
tb_alu.v alu.v
```

Set up the creation of a VCD file by placing VCD file generation statements in your testbench file. For example:

```
$dumpfile("_bluewav/iver.vcd");  
$dumpvars(0, tb_alu);
```

2. Set up and start *BlueHDL*.

Make sure that your PATH environment variable includes '/bluepc/bin', '/bluepc/gnu/bin/', and your tcl and Icarus Verilog paths such as '/progra~1/tcl/bin' and '/iverilog/bin'. Of course in Windows the '/' is replaced with '\\'.

Make sure that bwuser.tcl contains the correct settings for Icarus Verilog: the 'comptypewav' and 'simtypewav' Tcl variables should be set to 'iverilog' and the languagewav should be 'v'.

The bwuser.tcl file is also a good place to set the 'vcdfilename' Tcl variable. This value needs to match the name you used in the \$dumpfile command above. For example, if the \$dumpfile command specifies 'iver.vcd', you should set the 'vcdfilename' Tcl variable to 'iver'. Or you can set the 'vcdfilename' Tcl variable interactively in the Console Window. For the above example, you would enter: set vcdfilename iver (ret).

You can use one VCD filename such as 'iver' for all of your designs, if you don't want to keep changing this variable. Or of course you can use different VCD filenames for each of your designs, if you prefer.

One other Tcl variable may need to be modified for Icarus Verilog, the 'idelay' variable. This is the amount of time that *BlueWave* waits between the end of a simulation and the beginning of displaying the results. Its current default setting is 1000 (1 second). You may need to increase this value for large files because it may take a few seconds for the simulator to finish creating the VCD file. You can set this either in bwuser.tcl or interactively.

Start *BlueHDL* with bws for the Student Version (or bwp1 for the Pro 1 Version).

Left click in the Left Pane of the Compile Browser Window to change directories until you are in the directory containing your source code. If you followed the directions in Step 1 you should see a _bluewav directory in the Left Pane, and your files.v in the Middle Pane. The Right Pane isn't used with Icarus Verilog.

3. Compile your design using the GUI buttons in the Compile Browser Window. Either click on the Compile Button or double click on any source file.v in the Middle Pane to compile. This runs the 'iverilog -tvvp -o iver.vvp -c allfiles' command.

4. Simulate your design using the GUI buttons in the Waves Window.

Click on the Run Simulation Button. This runs the 'vvp iver.vvp' command. You should see waveforms appear in the Waves Window and signal names appear in the Navigator Window. There is no need to set up display signals first, *BlueHDL* automatically displays the test bench signals in the first simulation run.

Note that you won't see the hierarchy of your design in the Navigator Window. This is because Icarus currently does not employ the VCD \$scope command.

Use the Waves Window as described in [Tutorial 1 Zoom Functions](#) to examine the results by zooming and moving the line cursors. The simplest and most useful zoom commands are: Zoom in by drawing a Zoom Box with the Right Mouse Button. Zoom out by clicking on the Fit Button. Note: the Restart and Stop Simulation Buttons are not used with VerilogXL.

Using *BlueHDL* with SystemC

These are general instructions on how to use Blue Pacific's *BlueHDL* with SystemC running under Linux, Solaris and MS Windows. For more detailed instructions on running the Open SystemC simulator, see www.systemc.org.

SystemC is a C++ object-oriented, cycle-based simulator intended for hardware/software co-simulation. It is supported by Synopsys and about fifty other major semiconductor and electronics products companies. You can use Blue Pacific's *BlueHDL* Simulation tool to compile, simulate and display results, but you must first obtain the simulation kernel from www.systemc.org. After obtaining the simulation kernel from the SystemC Web site, you can use the examples they provide and follow the instructions below.

Steps for Running under Linux and Solaris Unix

1. Obtain the SystemC C++ code and documentation from SystemC. Go to www.systemc.org, obtain a password and download the files. Unzip and un tar (tar xvf) the files.

2. Configure SystemC.

cd to systemc area

```
./configure -- installs into /usr/local
```

```
make -- make
```

```
make install -- install
```

3. Modify existing examples or create your own to run under *BlueHDL*.

SystemC has a fairly complicated directory structure, and the easiest way to get up and running is to use the SystemC release examples directory. You can either use the the SystemC examples or modify and create your own files in their example directory with the following steps:

Make a `_bluewav` subdirectory in one of the SystemC release examples directories (`mkdir _bluewav`). Or create a new example subdirectory for one of your own designs and then create a `_bluewav` subdirectory under that.

Modify the `main.cpp` program to add VCD tracing to your design, use this code as an example:

```
// a clock similar to this would already be in the main.cpp file:
```

```
sc_clock clock("clock", 20.0, 0.5, 0.0);
```

```
// create vcd
```

```
sc_trace_file * tf = sc_create_vcd_trace_file("_bluewav/pipe");
```

```
sc_trace(tf, clock.signal(), "clock");
```

```
sc_trace(tf, in1, "in1");
```

```
sc_trace(tf, in2, "in2");
```

```
sc_trace(tf, sum, "sum");
```

Note that clock signals are unique and must be traced with the format of: `sc_trace(tf, clock.signal(), "clock");`

Modify the `main.cpp` program to add a standard run interface with argv

```
// control simulation
```

```
int n;
```

```
if (argc == 2) n = atoi(argv[1]);
```

```
else n = 200;
```

```
sc_start(n);
```

```
return 0; // prevent warning during compile
```

4. Use *BlueHDL* to compile, simulate and display the results.

Make sure that `bwuser.tcl` contains the correct settings for SystemC: the `'comptypewav'` and `'simtypewav'` Tcl variables should be set to `'systemc'` and the `languagewav` should be `'cpp'`.

Start *BlueHDL* with `bws` for the Student Version (or `bwp1` for the Pro 1 Version). Set the `'vcdfile'` Tcl variable to the VCD filename you used in `main.cpp`. For the pipe example above you would enter: `set vcdfile pipe (ret)`.

Left click in the Left Pane of the Compile Browser Window to change directories until you are in the directory containing your source code. If you followed the directions in Step 3 you should see a `_bluewav` directory in the Left Pane, and your files.cpp in the Middle Pane. The Right Pane isn't used with SystemC. You can then use either the GUI Buttons or enter Console commands to compile and simulate.

5. Customize the commands (if you like)

You can customize the functions of the buttons that control compilation and simulation as follows.

To modify the functions of the ReMake Button or other compile functions:

Open `$BLUEPC/usr/bwwin.tcl` with a text editor.

Search for `'Makefile.gcc'`. The first occurrence is in the `compile_hier` procedure, which is executed by the ReMake Button. The

second occurrence is in the `comp_com` procedure, which is executed whenever you double click on a source filename in Middle Pane of the Compile Browser Window.

Change this line to what you want.

If you need to modify the code that is called by the Run Simulation Button:

Open `$BLUEPC/usr/bwwin.tcl` with a text editor.

Search for 'run.x'.

Change this line to what you want.

Note: save a copy of `bwwin.tcl` before you edit it. Tcl/Tk is very picky about syntax and it's easy to create a syntax error that will cause `bwwin.tcl` not to load. We've also saved a copy of all the Tcl files in the `$BLUEPC/backup` directory.

Use the GUI Buttons to Compile and Simulate SystemC

Compile in the Compile Browser Window: Double click on any source file.cpp in the Middle Pane to compile (this runs the 'make -f Makefile.linux' command or 'Makefile.gcc').

Simulate in the Waves Window: Click on the Run Simulation Button to simulate for the run time value in the Run Time Box. You should see waveforms appear in the Waves Window and signal names appear in the Navigator Window. There is no need to set up display signals first, *BlueHDL* automatically displays the test bench signals in the first simulation run.

To change the run time, enter a new number in the Run Time Box, followed by a return. **** Note: Running the pipe example with more than 1000 ns of simulation will create not a number (nan) results and cause a crash ****. The Restart and Stop Simulation Buttons are not used with SystemC. Use the Waves Window as described here in the user's manual to examine the results by zooming and moving the line cursors. The simplest and most useful zoom commands are: Zoom in by drawing a Zoom Box with the Right Mouse Button. Zoom out by clicking on the Fit Button.

Use the Console Window to Compile and Simulate SystemC

You can use the Console Window to manually compile and simulate if you prefer command entry:

Compile and simulate in the Console Window:

Compile the design with:

```
make -f Makefile.linux (or use an alias)
```

Simulate the design with:

```
run.x runtime (where runtime is in nanoseconds)
```

```
example to run 400 ns: run.x 400
```

Display the results in the Navigator and Waves Windows:

In the Navigator Window:

```
File -> Load Sim Results File.vcd
```

Double click on the file.vcd to load the results

You should see signals appear.

In the Waves Window:

You should see waveforms appear.

Steps for Running under MS Windows

1. Obtain the SystemC C++ code and documentation from SystemC. Go to www.systemc.org, obtain a password and download the self-installing executable file.

2. Configure SystemC.

Double click on the self-installing file you just downloaded.

3. Modify existing examples or create your own.

See Step 3 in the instructions above for running SystemC under Linux and Unix. Make sure to create a `_bluewav` subdirectory in your source code directory.

4. Use *BlueHDL* to display the results.

Make sure that `bwuser.tcl` contains the correct settings for SystemC: the `'comptypewav'` and `'simtypewav'` Tcl variables should be set to `'systemc'` and the `languagewav` should be `'cpp'`.

You currently need to have MS Visual C++ Developer's Studio Version 6 or higher in order to use SystemC under MS Windows. Follow the SystemC instructions for compiling and executing SystemC files under Visual C++. If you followed the instructions in Step 3 you should see a VCD file created after running a simulation. The MS Visual C++ GUI makes it difficult to integrate compilation and simulation under *BlueHDL* but you can use it to display results. On the positive side, the Visual C++ GUI does provide a powerful debugging environment.

Use the Compile Browser to `cd` to your SystemC source code area. You should see a `_bluewav` subdirectory which contains the VCD results file.

Display the results in the Navigator and Waves Windows:

In the Navigator Window:

File -> Load Sim Results File.vcd

Double click on the file.vcd to load the results

You should see signals appear.

In the Waves Window:

You should see waveforms appear.

Other *BlueWave* Features

- [ATE files, mixed signal and analog simulation](#)
- [VCD \(Verilog Value Change Dump\) file compatibility.](#)
- [More about Tcl/Tk and C.](#)

ATE Files, Mixed-Signal and Analog Simulation

BlueWave supports both mixed signal simulation files and the generation of ATE files. It can display simulation results for both floating point numbers and standard vectors as analog waveforms. It also features translation procedures which generate test vector patterns for automatic test equipment manufactured by [Nextest](#). This feature provides a smooth flow from design to test. HDL test benches can be used to easily generate the files required for chip production with Nextest's line of VLSI test equipment.

VCD Compatibility

The *BlueWave* GUI allows the viewing of simulation results from other simulators, if they can create VCD (Verilog Value Change Dump) files. You can also use the Navigator Window 'Utilities -> Translate Mti Vcd to Bcd' Menu command to compress ModelSim generated VCD files.

To view VCD files, you first need to be in the correct directory. If you are only using *BlueWave* as a VCD viewer, you can set the `'comptypewav'` Tcl variable to `'viewer'` in `bwuser.tcl`, then use the built-in `cd` function of the Navigator 'File -> Load Sim Results File.vcd' Menu command to `cd`. If you are using *BlueWave* to control a compiler, the `'comptypewav'` Tcl variable will be set to `bvhdstud/pro1`, `fverilog` or `systemc`. In this case, use the Compile Browser Window to change directories to the directory above the `_bluewav` directory where your VCD files reside. When `'comptypewav'` is anything other than `'viewer'`, *BlueWave* always expects VCD results to be in a `_bluewav` subdirectory below your current directory.

After `cd`'ing to the correct directory, use the Navigator 'File -> Load Sim Results File.vcd' Menu command to load and display your results.

More About Tcl/Tk and C

The *BlueWave* GUI is a combination of Tcl/Tk and C. This provides a command-line interface shell (Console Window) with both standard Tcl/Tk commands and custom *BlueWave* commands. It allows the tool to be extended by the user while also creating the fastest possible performance since computationally-intense algorithms are coded in C.

Tcl/Tk is the brainchild of ex U.C. Berkeley professor John Ousterhout, who is currently at Scriptics, a company which he recently founded. It is a combination of a command shell and GUI development tool and it features a natural interface to C/C++.

Tcl/Tk is available for free from Scriptics, and it runs on all major operating systems. Scriptics currently offers a debugging and development environment which is useful for commercial developers, but not necessary for many users. The main Web site for Tcl/Tk is: dev.scriptics.com.

We have found both the online Tcl Help that comes with Tcl/Tk and several books to be extremely helpful in learning Tcl/Tk. The books are:

1. Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley Publishing, 1994, ISBN 0-201-63337-X.
2. Graphical Applications with Tcl & Tk, Eric Foster-Johnson, M&T Books, 1997, ISBN 1-55851-569-0.
3. Effective Tcl/Tk Programming: Writing Better Programs with Tcl and Tk, Mark Harrison and Michael McLennan, Addison-Wesley Publishing, 1998, ISBN 0-201-63474-0.

BlueWave External Source Code Editor and Printf-Like Debug

Source code editing is performed via the user's editor of choice. We provide you with an Edit Button in the Compile Window, but not an editor. You set up the Edit Button by setting the 'editcomwav' Tcl variable in bwsetup.tcl. We figured that everybody already has a favorite editor and that's what they should use for creating and modifying source code. We recommend using a VHDL/Verilog syntax sensitive editor like Vim (vi improved), available from www.vim.org or Emacs, available from www.gnu.org.

The *BlueHDL* Student and Pro 1 Versions do not provide a source-line debugger for code debugging. Instead, the user embeds Verilog \$monitor, \$display and \$write statements in the source code for both Verilog and VHDL (a slight but helpful extension of VHDL). These printf-like statements display information about signals and variables in the Console Window, and the user can usually easily see what is going wrong. Use the 'vlogsyswav' Tcl variable in bwsetup.tcl to turn this feature on and off.

Our experience has been that most problems are more easily debugged with a few well-placed printf-like statements rather than dealing with the extra complexity of a debugger. Many engineers use a source code debugger because many simulators don't display VHDL variables or even provide list file output. The *BlueHDL* tools provide both VHDL variable display and list file output.

Having said all this, the Pro 2 Versions of the tools will feature a source-line debugger.

Internal Data Structures Available to User

A key *BlueHDL* idea is that a main data structure showing the current values of signals/variables is available to you. The data structure is called finalwav and it is a Tcl list. You can access the current value for any signal/variable (the last value of the current simulation run) in the Tcl finalwav list data structure via an index.

To access data for the signal/variable of interest, use its index number which is listed in the Sigs/Vars Display List Pane. For example, using the final proc the user can retrieve the current value for the signal/variable with an index of 2 in the Navigator Window Sigs/Vars Display List Pane like so:

```
set myvalue2 [final 2]
```

Search for 'proc final' in bwwin.tcl to see how it accesses the finalwav list data structure.

BlueHDL Tools Are File-Oriented

The *BlueHDL* tools are a very file-oriented. The functions and appearance of the graphical user interface are controlled by Tcl/Tk files which are open and modifiable by the user. Also nearly all processes including compilation, simulation and display of waveforms generate and utilize files. These files are automatically generated and can often greatly assist in debugging problems.

The simulation output is file-based. Simulation results are placed in a BlueSim Change Dump (BCD) format file. The BCD file format is very similar to the VCD (Verilog Value Change Dump) file format. Both the BCD and VCD files contain all the

information related to a design and its simulation results in a nutshell in one file. Included in a BCD/VCD file are: design tree hierarchy, entity/module names, captured signal names with their formats and vector sizes, simulation results in a dump-on-change format. *BlueWave* has the capability to read standard VCD files, so it can also be used as VCD results viewer for simulation tools from other vendors.

The control of waveform display and output files is file-based. Most of the files of interest will be located in the `_bluewav` sub directory, under wherever you choose to work. The display order and format of signals and variables for all windows and output files is controlled by the file `_bluewav.dis` (located in the `_bluewav` directory). It is automatically generated each time the user runs a simulation or edits a bus format. The display format can easily be altered by using the Navigator Window or a text editor to modify either `bluewav.dis` or any `userfile.dis`.

Directory Structure

The bluepc sub directories are: `/backup`, `/bin`, `/examples`, `/html`, `/lib`, `/myhome`, `/usr`. In MS Windows there will an additional directory for GNU gcc, `/gnu`. The directory structures under the Linux, Solaris and MS Windows operating systems are similar. The bluepc install area can be anywhere. The install scripts will ask you where you want to install. The install script will modify your `autoexec.bat` or `.cshrc` to add two environment variables (BLUEPC and possibly DJGPP) and PATH items (`$BLUEPC/bin` and possibly `$BLUEPC/gnu/bin`).

The Scriptics Tcl/Tk distribution files will not be located under the bluepc install area. In Linux and Solaris they are usually located at: `/usr/lib/tcl8.3` and `/usr/lib/tk8.3`. In MS Windows they are usually located at: `/Program Files/Tcl`.

The `/backup` directory contains backup Tcl files, in case you accidentally add a hard-to-find syntax error when modifying a Tcl file.

The `/bin` directory contains all the executable files. This directory can be moved to a shared, central location when using Linux.

The `/examples` directory contains source code examples and is required for the tutorials.

The `/gnu` directory contains the GNU gcc C compiler. It will only appear under MS Windows and may be called `djgpp`. It will not appear under Linux because we assume that a Linux user already has gcc installed elsewhere.

The `/lib` directory contains library files for the C compiler.

The `/html` directory contains the BlueHDL html help files.

The `/usr` directory contains 4 of the 6 Tcl files: `bwsetup.tcl`, `bwhelp.tcl`, `bwproto.tcl`, `bwwin.tcl`. The remaining Tcl files, `bwuser.tcl` and `bwpref.tcl` are in the `/myhome` directory.

You will probably want to modify the `bwsetup.tcl` file. It contains initialization data that must be set up before windows are created and also data that won't be changed very often. Some colors, fonts and window pane ratios are set here. Also, the location of the `/bin` directory is here.

The `bwhelp.tcl` file contains basic help information. The `bwproto.tcl` file creates a user customized, product prototype window. We hope that you will modify the `bwproto.tcl` file. The `bwwin.tcl` file is the main GUI file and is about 6000 lines of code. You may want to modify the `bwwin.tcl` file, but only if you are an experienced Tcl/Tk programmer or want to learn this type of programming. It controls the functionality of all the windows. If you are more of a hardcore designer, you probably won't want to touch this file with a ten foot pole.

The `/myhome` directory is the default starting location for your working area. It also contains the master `bwuser.tcl` and `bwpref.tcl` files. These files allow you to customize the setup for each design. You may want to modify the `bwuser` file, because it controls features such as customized library mappings. Read through this file, it is self-documenting via comments. If you need special conditions for a particular design, you can copy these files from `/myhome` to the design area. *BlueHDL* starts in whatever directory you were in when you last exited the tool. It will use the local copy of `bwuser.tcl` in this startup directory if a local copy is present. The `bwpref.tcl` file controls many GUI features and the main window sizes and positions. You access Tcl variables in this file via the 'Edit -> Edit Display Preferences' Menu command in the Navigator or Waves Windows. You save new preferences with the 'Edit -> Save Display Preferences' command. The `bwuser.tcl` file loads the `bwpref.tcl` file.

Advantages and Disadvantages of Being File-Oriented

There are both advantages and disadvantages to being file-oriented. The advantages are primarily that the tools can be used with

other vendors' simulation and display tools, and that development and debug of the tools are greatly simplified. The simulator can output standard VCD format files. The GUI can read both standard VCD and VCDE format files. Both the simulator and GUI can also communicate via our own BCD format files.

The disadvantages are primarily in performance. Since both the simulator and GUI communicate via result files, the performance must be slower than a memory-oriented approach. However, as mentioned in the first section of this manual, the *BlueWave* GUI has been designed with several features that greatly enhance performance. If you are not generating List files and have the Rectangle Draw feature enabled, you should be happy with the performance of the GUI. The simulator itself also features good performance. It is within a factor of two in speed when compared with other top commercial simulators.

The Pro 1 Version features more control over memory allocation than the Student Version. The Student Version always allocates memory for 4k events per signal. The Pro 1 Version can either automatically allocate memory, or use a value provided by the user to allocate up to 1 million events per signal.

Automatic memory allocation is more efficient, but slower, since *BlueHDL* first counts all signal events to exactly allocate the correct amount. The 'eventmaxwav' Tcl variable is used to choose between automatic and user-controlled memory allocation for the Pro 1. If eventmaxwav is 0, *BlueHDL* automatically allocates memory based on event count. If eventmaxwav is > 0, *BlueHDL* allocates memory for eventmaxwav number of events. The eventmaxwav variable is in the bwpref.tcl file and can be changed and saved via the Navigator Window 'Edit -> Edit/Save Display Preferences' Menu commands. The future Pro 2 Version will feature both memory-oriented and file-oriented approach.

Console Commands

Test Commands:

```
showt show Navigator hierarchy tree data
testw1 test Waves with writeWave data structures
clearw clear Waves waveforms
```

General Compile, Simulation and Utility Commands:

```
bwexit exit simulator and save BlueWave info
comp fn or c fn compile VHDL/Verilog/SystemC filename fn
run or r run simulation for Run Time
runt t or rt t run simulation for time t
help this help info
? this help info
h command history
cdh cd home to homedir
cdu cd ../
pwc pwd and update current compile directory
final i get final value of sig/var i, example: final 3
gm map get logical to physical map, example: gm libmap
cdm map logdir cd to physical dir, example: cdm libmap work
```

Waves Window Commands:

```
left mouse place cursor C2 for Value at C2 function
right mouse zoom box
left arrow key snap to last change of sig/var under cursor
right arrow key snap to next change of sig/var under cursor

zi or i zoom in x 2
zo or o zoom out x 2
zf or f zoom fit
zex or x zoom expand
searchw search, see Tutorial 3 for search commands
s1, s2 or s3 search for 1, 2 or 3 values
sets t set Search Cursor at time t
dels delete Search Cursor
```

noc2 turn off Value at C2 function
snap i dir snap to dir change of sig/var i, example: snap 3 n
gethv get scrollbar info
seth .x set horizontal scrollbar at 0.x
setv .y set vertical scrollbar at 0.y

Error Messages, Known Bugs and Problems

The following are the main problems, error messages, and known bugs, along with their related solutions. The problems and errors reflect something wrong in a setup that can easily be fixed. The bugs listed here are either in Windows, Linux, Solaris or Tcl/Tk and cannot be solved by us, but we can suggest workarounds. *BlueHDL* bugs are listed in [bugs.html](#). For other bugs you encounter, send bug reports to bug@bluepc.com.

Problem:
When you try to run the *BlueHDL* simulator under Linux, the Console Window displays the error message: "Make: tb_file.exe: Command not found".

Solution:
You may need to install the latest version of *BlueHDL*. Earlier versions contained a bug that showed up in only some Linux Gnu C++ compilers. We have fixed this bug and verified the fix for Red Hat and Turbo Linux.

Or you may just need to add './' or '.' to your path to allow *BlueHDL* to find the simulation file in your current directory.

Problem:
When you try to run the *BlueHDL* compiler and simulator under Linux, the work library shows up in the Compile Browser Window as 'work\r' and a subdirectory named 'work?' appears in your source code directory. This can lead to several different error messages with the main result being you can't compile and/or simulate.

Solution:
You may need to install the latest version of *BlueHDL*. Some earlier versions contained some DOS format files in the Linux release that included the carriage return along with the line feed.

Problem:
When you try to run the *BlueHDL* simulator under Linux by typing 'bws' or 'bwp1', you get the following error message or something similar: "ld.so.1: bwstud: fatal: relocation error: file bwstud: symbol Tk_main: referenced symbol not found. Killed."

Solution:
You need to install Tcl/Tk 8.3. In order to be compatible with the largest number of Linux and Unix systems, *BlueHDL* for Linux and Unix use Tcl/Tk 8.3.

Problem:
When you run *BlueHDL* under Windows, the Console window appears in a different place on the screen each time.

Solution:
There is nothing you can do about this in Windows 95 and 98, but you can set up Windows NT and 2000 to control this. The problem is that Tcl/Tk can control all the windows except the Console Window through the 'wm' series of commands. In Linux this isn't a problem, because the script that calls *BlueHDL* (bws or bwp1) also controls the X term size and position.

In Windows NT and Windows 2000 you can control the Console Window with these steps: place the Console where you want it, then right click on the top of the Console Window and select the Properties Menu. Next size the window with the Layout choices and click OK. Select the 'Modify shortcut which started this window' option. The next time you start *BlueHDL*, the Console Window should appear as you saved it.

Remember that you can control all the other windows by placing and sizing them, then using the Navigator Window 'Edit -> Save Display Preferences' command.

Problem:
When *BlueHDL* tries to load, the OS presents a Dialog box with an Exception Error or only two *BlueHDL* windows come up.

Solution:
The solution is to solve the problem that is most likely related to the environment and/or path variables not being set correctly.

You need to change your Linux or Solaris shell setup (like `.cshrc`), or Microsoft Windows `autoexec.bat` (Windows 95/98) or `autoexec.nt` (Windows NT).

In Linux or Solaris enter 'printenv' in an X Window to check these variables. In Windows enter 'set' in a DOS Window to check these variables. You should see two environment variables that look something like (no C: and the slashes are forwards for Linux):

```
BLUEPC=C:\bluepc
DJGPP=C:\bluepc\gnu\gnu.env
```

You should see a path variable that looks something like: `PATH:=C:\bluepc\bin;C:\bluepc\gnu\bin;C:\progra~1\tcl\bin`

Description: this problem occurs when *BlueHDL* can't figure out where to find the Tcl/Tk and Gnu files it needs to run.

Problem:

When running *BlueHDL* for the first time or when running it with a new design directory, either a Tcl/Tk Error Dialog Box appears or the tool crashes completely.

Solution:

There are several possible causes and solutions to this problem.

One cause is that the directories where you are running *BlueHDL* are not open to writing, the `$BLUEPC/myhome` directory and subdirectories are usually the culprit. For example, you may have installed the software under Linux or Solaris as root and are trying to run as a normal user. The solution is to open up the bluepc directories for writing.

Another possible cause is related to Tcl/Tk timing problems when some initial files are being created. The solution is to run *BlueHDL* again. This problem occurs when Tcl/Tk tries to access some internal data in a new file it tried to create. The operating system could not create the new file either because of write permissions or because it didn't create the file in time for the data to be accessed.

Another possible cause is that *BlueHDL* can't find the shared libraries it needs. For Linux and Solaris you must ensure that the shared libraries required by *BlueHDL* are available. The solution is to set the `LD_LIBRARY_PATH` environment variable and/or adding symbolic links to existing shared libraries.

You should cd to `$BLUEPC/bin` and run `ldd` on all the files in this directory, for example: `ldd bwstud`. If you see any messages saying a library was not found, you need to find the closest library you have installed in `/usr/lib`, `/usr/local/lib`, `/usr/openwin/lib`, etc. and create a symbolic link (probably as root). For example, if `ldd bwstud` produced "libX11.so.6 not found" and you have libX11.so.4 installed, you need to cd to your X11 library area (probably `/usr/openwin/lib` or `/usr/X11R6/lib`) and create a symbolic link like so:

```
ln -s libX11.so.4 libX11.so.6
```

You will also need to ensure that Tcl/Tk Version 8.3 is installed and visible to *BlueHDL*. You may also need to create some symbolic links for Tcl/Tk. Under Linux and Solaris the easiest way to test Tcl/Tk is to enter 'which wish8.3' (or whatever Tcl/Tk release you have). Your system should tell you where the Tk shell wish8.x lives. Then you should be able to enter: 'wish8.3' and see a small Tk screen pop up.

Problem:

When *BlueHDL* loads, the windows are all piled on top of each other.

Solution:

Check the directory and file permission bits for any that are 'Read Only'. Remove any 'Read Only' permissions, and restart *BlueHDL*. In Linux, these are checked with 'ls -l' and set with the `chmod` command. In Windows these are checked and set by highlighting a directory, file or group of files, and then using the Right Mouse Button to click on Properties, then setting the Attributes.

Description: this problem occurs when *BlueHDL* can't copy and create files in its directories due to no write permission.

Problem:

There are no menu choices in the menu bar area at the top of the windows.

Solution:

Edit the `$BLUEPC/usr/bwsetup.tcl` file and search for 'set newmenu'. Change the line from 'set newmenu yes' to 'set newmenu

no'.

Description: this problem occurs with some versions of Windows 98. A new syntax for creating menus is available in Tcl 8.3 Versions and higher, but this syntax doesn't work with some versions of Windows 98.

Problem:
The text in menus or buttons is garbled or incomplete.
Solution:
Edit the \$BLUEPC/usr/bwsetup.tcl file and search for 'font'. Experiment with creating new fonts or using some standard X fonts. One user reported that using this line cleaned up a Button text problem in Linux:
set butfontwav "-adobe-times-medium-r-normal--10-120-75-75-p-64-iso8859-1"

Description: this problem occurs with some versions of Linux or Unix and depends upon your screen resolution and system fonts.

Problem:
When you use the file menu choice to open a file, nothing happens. For example, if you use the Navigator Window 'File -> Load Sim Results File.bcd (or vcd)' command and no waveforms appear in the Waves Window.
Solution:
Change the filename from upper case to lower case or type in the filename in the 'File name:' field.

Description: this problem occurs with some versions of Windows 98, NT and 2000 where the Tcl/Tk file open command doesn't deal properly with upper case.

Problem:
** Error: can't open file.bcd.

** Error: can't open file.dis.

** Error: can't open file.lis.

** Error: can't open file.sim.

** Error: can't open file.sig.

You may need to use the Compile Browser Window to cd to a directory first, OR you may need to remove write protection for directory or file.

Solution:
As the error message states, the problem may be that you didn't use the Compiler Browser window to cd to the directory that contains the file. While you can cd in the File Open Dialog Box, this doesn't change the directory for *BlueWave* itself.

Or the problem may be the file permission bits. Check the directory and file permission bits for any that are 'Read Only'. Remove any 'Read Only' permissions, and restart *BlueHDL*. In Linux, these are checked with 'ls -l' and set with the chmod command. In Windows these are checked and set by highlighting a directory, file or group of files, and then using the Right Mouse Button to click on Properties, then setting the Attributes.

Description: this problem occurs when *BlueWave* can't read, copy or create files in its directories due to being in the wrong directory or due to lack of write permission.

Problem:
** Error: file.dis is not a subset of file.sig. Create a new file.dis:
Solution:
1. click on Navigator Clear Button.
2. click on Waves Run Simulation Button,
OR click on Navigator Update Waves Button.

Description: this error occurs when the Navigator Sigs/Vars Display List Pane is out of date with respect to the simulation results file, which then causes a mismatch between the display file, _bluwav.dis, and the complete signals/variables list file, design.sig. It is usually due to adding or deleting signals/variables in your design, but still using an old display list. As stated in the error message, the solution is to clear out the Sigs/Vars Display List Pane using the Clear Button, and then either rerun the

simulation or update the waves with an empty Sigs/Vars Display List Pane (that is, automatically create a new file.dis using the Display Level).

Problem:

** Error: can't load a display file when file.bcd Box is empty.

Solution:

Either run a simulation to automatically map file.bcd

OR use the Navigator 'File -> Load Sim Results from Menu' command.

Then load display file.

Description: this error occurs when the Navigator Window design tree hierarchy information contained in file.sim has not been loaded. As stated in the error message, the easiest solution is to load simulation results from a previously existing file.bcd, which will also force a load of file.sim. Then you can load a display file (file.bcd).

BlueHDL Downloading and Installation

The Student Versions of VHDL, SystemC and foreign Verilog running under Linux, Solaris and Microsoft Windows are all available for downloading now.

[Click here](#) to download any of our Student Version files.

Linux and Solaris Downloading and Installation

BlueHDL under Linux was debugged using Red Hat 6.0. It should also work under other versions of Linux, but may need additional symbolic links. *BlueHDL* under Solaris was debugged using Sparc Solaris 2.7 on a Sun Workstation. For Linux and Solaris we make the assumptions that you already have Tcl/Tk and GNU gcc loaded and working. If not, you should get the latest Tcl/Tk Version from Scriptics and the latest gcc from GNU (see the links below). For Linux and Solaris Tcl/Tk needs to be Version 8.3.

Once you have downloaded the Linux or Solaris file gunzip the file with: gunzip file_tar.gz.

Tar extract the unzipped file with: tar xvf file_tar.

Run "setup" at the top level CD directory or unzipped directory. See the README file in the unzipped directory for more details.

If You Have Problems with Linux or Solaris

If *BlueHDL* doesn't load, the required environment variables may not be set correctly. To verify that these are set correctly, enter 'printenv' and a return with no arguments in a window. Assuming that /home/yourname/bluepc is your install path, you should see the following environment variables:

```
BLUEPC=/home/yourname/bluepc
```

```
PATH=./:/home/yourname/bluepc/bin:otherpaths
```

We have installed *BlueHDL* on several different Linux releases and several Solaris releases. After the environment variables are set up, the main problems come from shared libraries not being visible. See either [Tutorial 1](#) or the [Problems Section](#) for details on solving these problems

Relevant Web sites: Red Hat: www.redhat.com. Scriptics: dev.scriptics.com. GNU: www.gnu.org.

Microsoft Windows Downloading and Installation

BlueHDL under MS Windows was debugged using Windows 95, 98 and NT. For Windows we make the assumption that you have neither Tcl/Tk nor GNU gcc loaded, both are required and included in the download file.

To install the free student version of BlueHDL from CD-ROM, insert the CD into your CD-ROM drive. Or if you downloaded the latest version from our Web site, unzip the WINZIP file. Then run "setup" at the top level directory on the CD or under the unzipped directory. Setup will ask you several questions and then execute. Once the installation is finished, drag and drop the

BlueHDL Icon from the \$BLUEPC\bin directory onto your desktop. Then you are ready for Tutorial 1 in the User's Manual: \$BLUEPC\html\blueuser.htm.

If You Have Problems with MS Windows

If *BlueHDL* doesn't load, see the README file in the unzipped directory. The required environment and path variables may not be set correctly. To verify that these are set correctly, enter 'set' and a return with no arguments in a DOS window. Assuming that C: is your install path, you should see the following two environment variables and three entries in your PATH variable:

```
BLUEPC=C:\bluepc
```

```
DJGPP=C:\bluepc\gnu\gnu.env
```

```
PATH:=C:\bluepc\bin;C:\bluepc\gnu\bin;C:\progra~1\tcl\bin
```

If C: is not your installpath, the only additional change you need to make is to change the Properties of the BwStud shortcut to point to your installpath. For more details, read the README file.

**** NOTE1: Be sure to reboot your computer after installation, so the new environment and path variables can take effect.

**** NOTE2: *BlueHDL* gets confused if you install it in the same directory where you unzipped bhstud.zip, so you need to install it in a fresh directory. We've also had some problems when we've installed in lower level directories, so you may save yourself some grief if you install either in C:\ or in D:\.; ie, your environment variables should look like the examples above.

Unsupported IEEE VHDL 1076 Features

We are still lacking some features and are working to bring *BlueHDL* VHDL up to the IEEE VHDL 1076 Spec. See [ieee1076.html](#) for details.

Support

We provide the best level of support that we can given the small size of Blue Pacific. We answer email questions within a few days and phone questions immediately when we are in the office. Much of the time we are out of the office consulting or teaching.

We are currently providing a higher level of support for a few customers. If you are working for a company with an interest in purchasing ten or more licenses, we would be happy to add you to this group.

Contact us at: (858) 484-7500

Support questions email: info@bluepc.com

Bug report email: bug@bluepc.com

Hardware Requirements

The following are the minimum hardware requirements for *BlueHDL*.

- CPU: Pentium class
- RAM: 32 Megabytes
- Hard disk: 25 Megabytes

Index

- [Bus Format Properties](#)
- [Color Properties](#)
- [Commands](#)
- [Compilers](#)
- [Customization](#)
- [Downloading and Installation](#)

- [Edit Display Preferences Menu](#)
- [Edit Signal/Variable Properties](#)
- [Error Messages, Known Bugs and Problems](#)
- [Files and Directory Structure](#)
- [Foreign Verilog](#)
- [Graphical User Interface](#)
- [Help](#)
- [High-Speed Waveform Display](#)
- [Icarus Verilog](#)
- [Labels](#)
- [Mixed Signal Simulation](#)
- [Name Search](#)
- [Search Names](#)
- [Search Waves Window](#)
- [Signal/Variable Properties](#)
- [Simulator](#)
- [SystemC](#)
- [Tcl/Tk](#)
- [Tutorial 1](#)
- [Tutorial 2](#)
- [Tutorial 3](#)
- [VCD File \(Verilog Value Change Dump\)](#)
- [VCD File Translation](#)
- [VHDL, Verilog and SystemC Compilers](#)
- [Windows:](#)
- [Compile Browser Window](#)
- [Console Window](#)
- [List File Window](#)
- [Navigator Window](#)
- [User Proto Window](#)
- [Waves Waveform Window](#)
- [Zoom Functions](#)

*Cadence VerilogXL, Mentor ModelSim and Open SystemC are registered trademarks of their respective companies and organizations.

*Avanti HSpice and Cadence PSpice are registered trademarks of their respective companies.

[return to top](#)
