WaveFormer Pro

VeriLogger Pro

Timing Diagrammer Pro

User's Manual

www.syncad.com

Timing Diagrammer Pro, VeriLogger Pro, WaveFormer Pro Manual (rev 6.5A) copyright 1994-1999 SynaptiCAD

For latest product information and updates contact SynaptiCAD at:

web site: http://www.syncad.com

email: sales@syncad.com

phone: (540)953-3390

Table of Contents

Introduction

The quickest way to learn how to use this product is to read the **Quick Overview** section and to work through the tutorials that start on page 163 and are included with the on-line help. Our package is relatively easy to use, but it is easy to miss some of the more powerful features if you skip over the tutorials. If you have previously used other timing diagram editors then you should pay close attention to the sections on formulas, library-macro features, buses, equation based waveform generation, VHDL-Verilog stimulus generation, and the interactive HDL simulator features.

Before you start a real design, read over the check list in the **Starting a New Design** section to make sure that you are ready to go. Correctly setting up a project only takes a few minutes and it can save you a lot of time in the end.

The body of the manual is organized so that each chapter contains all the actions about a particular object. For instance, if you want to find out how to use formulas with clocks then you should look in the chapter on clocks. The on-line help contains the most up-to-date list of features and functions (including the entire text of the manual).

Quick Overview

TestBencher Pro, VeriLogger Pro, WaveFormer Pro, and Timing Diagrammer Pro are all based on the same object-oriented timing diagram editor, so drawing and editing commands are the same across products. Chapters 1, 2, and 3 cover the basic drawing of waveforms, signals, clocks, and buses. The basic drawing and editing commands are:

- Use the **Add Signal** or **Add Clock** buttons to add signals and clocks.
- Use the **left mouse button** to draw a waveform segment from the end of the signal to the mouse cursor. The type of waveform drawn is determined by the red **state button** (HIGH, LOW, etc.). The state buttons automatically toggle between the two most recently activated states. The state with the small red "T" above the name will be the toggle state.
- **Double left click** on an object to edit the object's properties.
- **Drag-and-Drop** to move objects.

Chapters 4, 5, and 6 cover the basic timing analysis features which are performed by placing parameters like delays, setups, holds, and samples on the timing diagram. These parameters define the causal relationships between signal edges. The **right mouse button** is used to add parameters, text objects and markers. To add a timing parameter:

- Left click on one of the mode buttons **Delay**, **Setup**, **Hold**, **Sample**, **Marker**, or **Text** to activate a mode (the button will turn red).

- In one of the parameter modes (delay, setup, hold, or sample), **left click** on one transition to select it, **then right click** on the second transition to add a parameter between the first and second transitions.

- In text mode or marker mode, click the right mouse button to add the object.

Chapter 12 covers the advanced timing analysis which is supported through the Interactive HDL Simulation engine included with WaveFormer Pro, VeriLogger Pro and TestBencher Pro. The Interactive HDL simulator supports behavioral HDL code, Boolean equations, and true min/max simulation. Changes in the input waveforms automatically trigger resimulations. The simulation features are reached through the *Signal Properties* dialog:

- Double left click on a signal name to open the *Signal Properties* dialog.

- Use the **Boolean Equation** or **HDL code** edit boxes to define the simulation for the signal.

- Check the **Simulate** radio button to continuously simulate the signal.

Chapters 7, 8, and 9 cover image, printing, and timing diagram formatting features. All three products provide six different methods for embedding timing diagram images into documents and page layout programs. All vector-based image files are generated using **File > Print** menu commands and include: WMF, CGM, & EMF metafiles, FrameMaker MIF, and EPS with previews. The **Edit > Copy-to-Clipboard** provides a Windows bitmap image of the current window and is compatible with almost all Windows programs. Also supported is the new industry standard TDML format and OLE.

The rest of the chapters are dedicated to helping WaveFormer Pro, VeriLogger Pro and TestBencher Pro interface to the rest of your EDA tool suite:

- Chapter 11 covers temporal equation based waveform generation and state labeling equations.

- Chapters 13 and 14 cover import and export of waveform data. Over 20 different formats are supported including Verilog, VHDL, SPICE, and HP logic analyzers. Support for new import and export formats can be written by the user using the built-in Perl script interpreter.

The timing diagram editor is an object-based drawing environment. This means you will be able to extrapolate most functions if you know how to do something similar. For instance, one way to delete a signal is to select it by left clicking on the signal name then pressing the delete key on the keyboard. Similarly, to delete a delay object, select it with the left mouse button and press the delete key.

The Table of Contents will help you locate more advanced functions. Take some time to draw signals, press all the buttons, look through all the menus, and work through the tutorials. Before beginning a real design you should read **Starting a New Design** in order to optimally set up your project files.

Starting a New Design

When starting a new design, it is important to set up your project correctly. Before designing, we recommend the following actions:

1) Determine the **Display Time Unit** and **Base Time Unit** for the current project. The **Display Time Unit** sets the units for entering and displaying time values. The **Base Time Unit** controls the time resolution of the design, and it is usually an order of magnitude lower than the Display Time Unit. If you normally use nanoseconds, then a good Base Time Unit is picoseconds. The next section describes Base and Display Time Units.

2) Decide if you will be using a formal library structure.

- If you are using a formal library structure, then decide on macro and library names, configure the library search list, and make your libraries. Chapter 10 and the Parameter Libraries tutorial describe this procedure.
- If you are not using a formal library structure, then decide what types of free parameters you will need to represent common parts in your design. Either type these in or extract them from older projects using the **Libraries > Save Free Parameters As Library** menu option. Chapter 5 covers the use of free parameters.

3) Add all clocks with correct period and duty cycle. If you encounter clock frequency rounding errors then lower the Base Time Unit.

4) Add everything else (signals, parameters, etc.).

Base and Display Time Units

The program has a limit of approximately three trillion time units (varying with screen resolution and size). The **Base Time Unit** sets the smallest amount of time that can be represented in the program. The **Display Time Unit** sets the units of all times displayed and entered.

Normally the base time unit is a magnitude smaller than the display time unit. This gives you the ability to specify exact values while allowing most other numbers to be entered at a convenient level.

To set the Base Time Unit:

- Choose the **Options > Base Time Unit** menu item. This will display the *Base Time Unit* dialog box.
- Select the new base time unit by clicking the proper radio button (fs, ps, ns, us, ms).
- Next, decide how the existing values in the project will convert to the new base time unit. For example, if you change from ns to ps then: **Keep All Times** makes an existing 5ns into 5000ps (the same time value); **Scale All Times** does not change the numbers but it changes their meaning (5ns will now become 5ps); **Scale Parm Times, Keep Signal Times** is a combination of the other two methods.

- Click the **OK** button to make the changes.

To set the Display Time Unit:

- Choose the **Options > Display Unit** menu item. This will display a submenu of display time units. The checked time is the current Display Time Unit. (Default is ns = nanoseconds).
- Click on a time unit to make it the new Display Time Unit.

Chapter 1: Signals and Waveforms

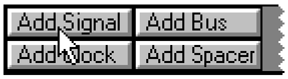
Signals and drawn waveforms are the simplest form of signals that make up timing diagrams. Later chapters cover more advanced signal types which calculate their own waveforms or display specialized information:

- **Clocks**, repetitive waveforms that draw themselves, are covered in *Chapter 2: Clocks*.
- **Group Buses** and **Virtual Buses**, signals which display the values of their member signals, are covered in *Chapter 3: Group Buses and Virtual Buses*.
- **Simulated Signals** which calculate their waveforms based on Boolean equations, registered logic, and behavioral HDL code are covered in *Chapter 12: Interactive HDL Simulation*.

The following functions cover adding signals, editing signal properties, and drawing waveforms.

1.1 Adding and Deleting Signals

To add a new signal:



- Left mouse click on the **Add Signal** button.

To delete a signal or multiple signals:

- Select the signal names by left clicking on them. A selected signal's name is highlighted.
- Then press the **delete key** on the keyboard or select the **Edit > Delete** menu option.

Note: When a signal is deleted, any parameters or text connected to it are also deleted. If you make a mistake deleting, you may undo the delete by choosing the **Edit > Undo** or **Edit > Undo Delete** menu item.

1.2 Editing Signal Properties

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

- Double left click on the signal name to open up the *Signal Properties* dialog.
- Enter the new data and click **OK** to close the dialog.

Note: The properties of several signals can be edited at the same time by selecting the signals then right clicking in the *Label* window and select the menu option **Edit Selected Signal(s)** from the context menu.

There are four sections in the *Signal Properties* dialog box (See Figure 1.1 on the following page):

1) **Name Section:** There will be a valid default name already in the edit box. Names must be at least one character long (no spaces are allowed). Signal names beginning with "**$$**" are reserved for in-

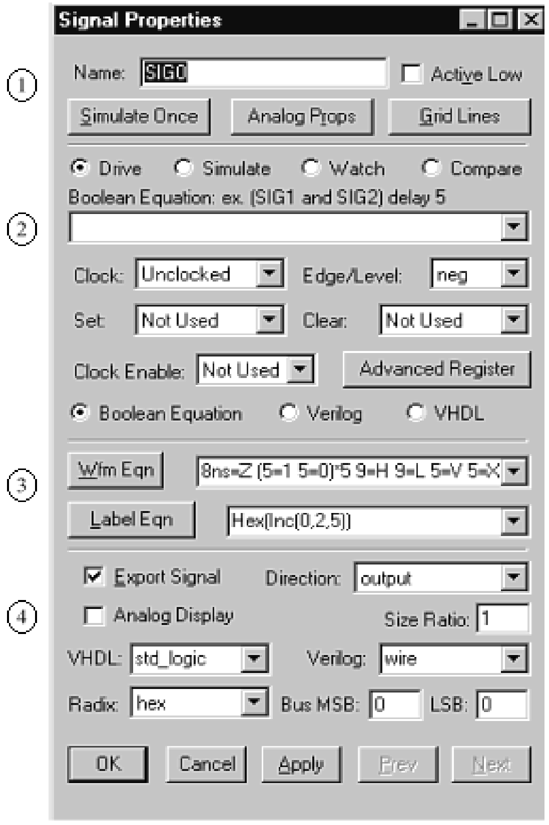
ternal use. Signal names should not begin with a digit or contain mathematical operators. Signal names should not be the same as common Boolean operators.

Active Low Name Checkbox: If checked, the name. of the signal is appended with a **$BAR** and the signal is displayed with a bar (line) drawn over the name. If an active low signal is used in the equation of another signal, then the name of the active low signal is **name$BAR**.

Analog Properties button: Opens the *Analog Properties* dialog which lets you set the logic voltage levels and rise/fall times used by the SPICE output feature. There is also a **Properties** button that opens the general *Object Properties* dialog box. This is an advanced feature that is available to users writing Perl scripts to support the storage/editing of user-defined attributes attached to WaveFormer objects. For more information read Chapter 14: Object Properties.

Grid Line Button: Opens the *Grid Options* dialog, which specifies how to draw vertical lines from the signal edges.

2) Logic Wizard Section: In WaveFormer Pro, VeriLogger Pro and TestBencher Pro, a signal can be described as a registered Boolean equation of other signals. Specify the Boolean equation, timing information, and register type using this dialog. Click on the **Simulate Once** button to see the results. Choose the **Simulate** radio button for automatic resimulation. The **Drive** state is the default state for signals and is used to describe signals that are drawn by the user. The **Watch** state is used to view signals embedded in external models. For more information read *Chapter 12: Interactive HDL Simulation*


Signal Properties

1 Name: SIG0 ☐ Active Low

Simulate Once | Analog Props | Grid Lines

[✓ Drive] Simulate Watch Compare

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

Clock: Unclocked Edge/Level: neg

Set: Not Used Clear: Not Used

Clock Enable: Not Used | Advanced Register

[✓ Boolean Equation] Verilog VHDL

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

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

☑ Export Signal Direction: output

4 ☐ Analog Display Size Ratio: 1

VHDL: std_logic Verilog: wire

Radix: hex Bus MSB: 0 LSB: 0

OK | Cancel | Apply | Prev | Next

Fig. 1.1 Signal Properties Dialog Box

3) **Temporal Equation Section:** A signal's waveform can be generated using a Temporal Equation and automatically labeled using a Label waveform equation. For more information read *Chapter 11: Waveform Equation Generation.*

4) **Export Properties Section:** These properties affect how WaveFormer Pro, VeriLogger Pro and TestBencher Pro export signals. They also affect how a signal's values are displayed.

- **Export Signal** check box determines whether or not a signal will be exported to stimulus and test bench files.
- **Analog Display:** This checkbox is used to determine is the selected signal will be displayed as an analog signal or not.
- **Direction:** Used for exporting VHDL or Verilog code. See the TestBencher pro manual for more information.
- **Size Ratio**: Used to determine the display height of the signal.
- **VHDL:** and **Verilog:** Determine the HDL type of the signal (e.g. std_logic, integer, wire,...). The user can also enter a new type into the drop-down edit box. The VHDL and Verilog history lists can be rearranged by editing the **ini** file (**timing.ini** for Timing Diagrammer Pro, **waveform.ini** for WaveFormer Pro, and **tbench.ini** for TestBencher Pro) located in the Windows directory. Be sure the program is not running when you edit its **ini** file.
- **Bus MSB and LSB:** Determine the bit size of the signal. A single bit signal is (0,0). This is used by the Interactive HDL Simulator to determine the result of multi-bit operations. This is also used during the export of VHDL or Verilog stimulus files and test benches to determine how to initialize the signal in the HDL model.
- **Radix:** Determines the base in which signal values are displayed.

1.3 Drawing Waveforms

The timing diagram editor is always in drawing mode. To draw the waveform of a signal:

- Place the mouse cursor inside the Diagram window at the same vertical row as the signal name.
- Press down on the left mouse button. This draws a waveform from the end of the signal to the mouse cursor. The type of waveform drawn is determined by the red **state button** on the button bar (see below for more information on the state buttons).
- Move the mouse to the right and click again to draw another segment.

The **state buttons** are the buttons with waveforms drawn on their faces located on the button bar: HIGH, LOW, TRIstate, VALid, INValid, WHI weak high, and WLO weak low. When a state button is activated, it is pushed in and colored red. The active state will be the type of waveform that is drawn next. To activate a state button, left click on it.

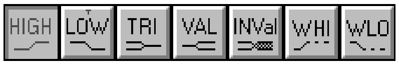


Figure 1.2. State buttons.

The state buttons automatically toggle between the two most recently activated states. The state with the small red "T" above the name will be the toggle state. The initial active state is HIGH and the initial toggle state is LOW.

When you draw signals using the mouse, the signal edges are automatically aligned to the closest edge grid time. The edge grid can be controlled from the **Options > Grid Settings** menu item.

1.4 Editing Waveforms with the Mouse

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

1) **Drag-and-Drop a signal transition:**

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

2) **Drag-and-Drop groups of signal transitions using <Shift> and <Ctrl>:**

- Hold down <Shift> or <Ctrl> while dragging a transition. This causes all the transitions to the right or left (respectively) of the selected transition to move with the selected edge.
- **Note:** Holding down both keys causes all the edges on the entire signal to move.

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

3) **Click-and-Drag to insert a segment into a waveform:**

- Left click in the middle of a wide segment (for this operation to work the segment must be wide enough to be selected).
- Drag to the right and then release the mouse. A new segment will be added in the middle of the original segment.

4) **Change a segment's graphical state by selecting it and then pressing a state button:**

- Left click in the middle of the segment to select it.
- Left click on a state button to apply that graphical state to the segment. State buttons are the buttons with the waveforms drawn on them (see Figure 1.2).
- **Note:** If you change the level of the segment to the same level as an adjacent section, the transition between them will turn red. This transition can then be deleted if necessary.

5) **Delete a segment by selecting it and then pressing the Delete key:**

- Left click in the middle of the segment to select it.

- Press the delete key on the keyboard.

These techniques will only work on signals that are drawn. They will not work on generated signals like clocks and simulated (blue) signals that are covered in later chapters.

1.5 Editing Waveforms using Dialogs

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

1) Moving and Locking Edges: Edges can be moved or locked using the *Edge Properties* dialog. *Chapter 5: Timing Analysis and Time Formulas* discusses the timing analysis features of this dialog. To edit a signal's edge:

- Double click on an edge of the signal transition to open the *Edge Properties* dialog.

- To move an edge, enter a new **min** or **max** time then click the **OK** button.

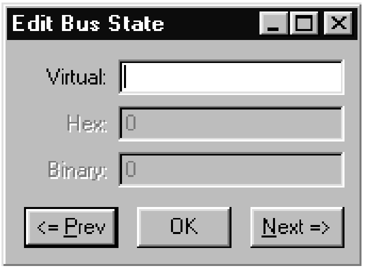
- To lock an edge so that it cannot be moved, check the **Locked** checkbox.

2) Adding a Virtual state to a Segment: In addition to the seven graphical states, a virtual data value can be added to a segment using the *Edit Bus State* dialog. To add a virtual state to a segment:

- Double click on the middle of a segment to open the *Edit Bus State* dialog.

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

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



1.6 Editing Waveform Edges from an Equation

You can edit the edges of a given signal or group of signals using the *Edit Waveform Edges* dialog. This feature works for all signals except clocks (use the *Clock Properties* dialog to edit these) and group buses. All of the edges of the selected signals that fall within a specified time range will be edited according to the equation provided. By default, the time range begins at the beginning of the diagram and ends with the last signal edge.

To Edit Waveform Edges:

- Select the **Edit > Edit Waveform Edges** menu option. This will open the *Edit Waveform Edges* dialog.
- Select the signals you wish to modify in the diagram window. Note: If no signals are selected, all signals will be modified except clocks and group buses.
- Specify the time range to modify using the **From:** and **To** edit boxes.
- Enter the equation to be used for waveform modification. The variable that represents the current time for each edge is **$time**. An example equation is:

  ```
  ($time + 10)/(4)
  ```

- Press the **Apply** button to move the waveform edges. This will leave the *Edit Waveform Edges* dialog open for other waveform modifications.

OR

- Press the **OK** button to move the waveform edges and close the *Edit Waveform Edges* dialog.

1.7 Moving Signals (Drag and Drop)

To move one or more signals:

- Left click on the signal names to select them. **Note:** The order that the signals are selected will be their new order after the move.
- Put the mouse cursor near the very top or bottom of any of the selected signal's names. When the mouse cursor changes from a normal arrow to an up/down double arrow, click down and hold the left mouse button. A green bar will appear.
- Drag the green bar to the signal's new location.
- Drop the green bar by releasing the left mouse button.

1.9 Hiding and Showing Hidden Signals

Hiding a signal removes the signal and any attached parameters and text from the screen, but not from the project. Delays, setups, and holds will continue to function, but timing errors will be shown only in the parameter window.

There are two methods that can be used to hide signals. The first method works by hiding selected signal names. The second method works by using filter patterns to match against the signal names (this method is particularly useful if your project has hundreds of signals). Each hiding method operates independently so both methods can be used to hide different signals at the same time.

1) To hide or show one or more signals by selecting names:

- Left click on the signal names to select them.
- Select the **View > Hide Selected Signals** menu option to hide the signals.
- Use the **View > Show Hidden Signal** menu option to show signals that were hidden using this method. This menu brings up a dialog box that allows you to selectively unhide signals.

2) To hide or show signals using filter patterns:

- Choose the **View > Filter Signals** menu option to open the *Signal Filter* dialog. This is a modeless dialog that can be left open while you work with the diagram.
- Choose either the **Show Only** or the **Hide** radio button to determine how the filter patterns act. Note: if at some point you wish to see all the signals that are hidden by the patterns, choose the opposite radio button.
- Enter a filter pattern in to the **Pattern** edit box. For example, the pattern **CLK** would match signals named **CLK0** and **CLK1**.
- Press the **Add** button to activate the pattern. Multiple patterns can be entered.
- To delete a pattern, select it and press the **Delete** button.

Related Topics: Selectively hide all delays, setups, holds, or text by unchecking the appropriate menu item in the View Menu. It is possible to hide individual parameters by selecting the parameter and then choosing the **View > Hide Parameter** menu item.

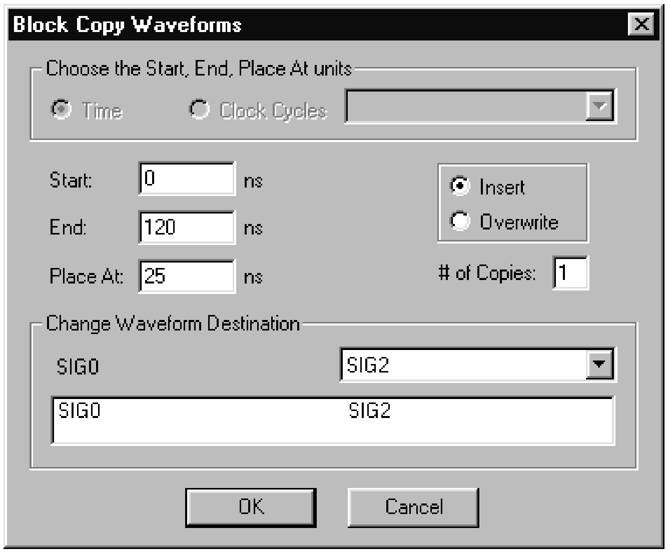
1.9 Block Copy of Waveforms

You can copy and paste sections of waveforms either onto (overwrite) or into (insert) any signal in the diagram. When a waveform section is copied any attachments in that section are also copied (text objects, delays, samples, etc.).

To copy and paste waveform sections:

- Select the names of the signals that you want to copy. If no signals are selected, the Block Copy command will select all the signals in the diagram (this is the most common usage).
- Choose the **Edit > Block Copy Waveforms** menu option. This opens the *Block Copy Waveforms* dialog box (see Figure 1.3), with the selected signals displayed in the Change Waveform Destination list box.
- In the dialog, enter the values that define the copy and paste:
    - Choose either **Time** or **Clock cycle** for the base units of the dialog. If you are copying just signals (no clocks) then **time** is the default base unit of the dialog. If you are copying part of a clock then it is best to choose a **clock cycles** base unit and choose the copied clock as the master clock. If you choose **time** when copying clocks the

(*end_time - start_time*) must equal an integral number of clock periods, and the *place_at* time must be at the same clock period offset as the *start_time*.



Block Copy Waveforms

Choose the Start, End, Place At units: [✓ Time] Clock Cycles

Start: 0 ns

End: 120 ns

Place At: 25 ns

[✓ Insert] Overwrite

# of Copies: 1

Change Waveform Destination

SIG0 SIG2

SIG0 SIG2

Figure 1.3 Block Copy Waveform dialog box

- **Start** and **End** define the times of the block copy.

- **Place At** is the time that at which the block will be pasted.

- The **Insert** and **Overwrite** radio buttons determine whether the paste block will be inserted be into the existing waveforms or overwrite those waveforms.

- The list box at the bottom of the dialog determines which signal the copied waveforms will be pasted into. To change this mapping:

  1) Select a line in the list box. This places the destination signal into the drop-down list box on top of the list box.

  2) Choose another signal from the drop-down list box. Each destination signal can be used only once per copy.

- Click **OK** to complete the copy and paste operation.

Alternate Method: The Block Copy **start** and **end** times can be selected using the mouse:

- Place the mouse cursor in the timeline in the Diagram window at the time where you wish to begin the block copy of signals.

- Press and hold the **<Ctrl>** key while clicking and dragging the mouse to the time where you wish to end the block copy of signals.

- Release the mouse then the <Ctrl> key to open the *Block Copy* dialog with the start and end times already entered in the dialog.
- Adjust the rest of the controls and click OK to complete the operation.

The Block Copy command is very powerful and can be used to perform many different functions such as:

- **Copy Diagram Cycles:** When copying read or write cycles with parameters attached to clocks, the best results are achieved if the (*end_time - start_time*) equals an integral number of clock cycles.
- **Insert time into the middle of an existing diagram:** Add some new signals (with no waveforms) and block copy the new signals into the existing diagram. The end time of the blank signals defines the width of the inserted space.
- **Copy parameters attached to clocks:** When performing the copy, select the clock and the signals that the parameters are attached to. Also define the copy block to be one or more complete clock periods.

1.10 Copying and Pasting Signals

You can copy and paste signals to another place in the current timing diagram or to another timing diagram by using the clipboard as a temporary holding place.

To copy signals:

- Select the signals by left clicking on the signal name(s).
- Choose the **Edit > Copy Signals** menu option or the **<Ctrl>-C** keyboard shortcut. Now the signals and any objects attached to the signal (text objects and parameters for example) are copied to the clipboard.

To paste the signals back into a timing diagram:

- Choose the **Edit > Paste Signals** menu option or the **<Ctrl>-V** keyboard shortcut. When the signals are read from the clipboard they obey the same rules as if they were being merged from a different file.

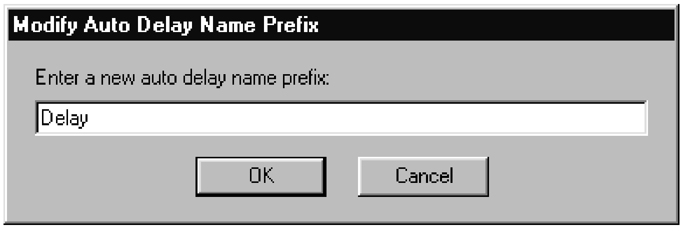
Note: If you want to a copy a portion of the waveforms (a time slice) on one or more signals, use the **Edit > Block Copy Waveforms** menu option. If you would like to add a piece of waveform onto the end of a signal, then the Temporal Equation feature can be used to do this. Also, if you want to shift some or all the edges in a signal then press **<Ctrl>** or **<Shift>** while dragging an edge.

1.11 Changing the Auto Name Generation Settings

Signals, clocks, buses, delays, setups, and holds will be assigned a default name made up of a prefix and a number when they are created (SIG, CLK, BUS, D, S, H).

To change the prefix used:

- **Right mouse click** on the mode button or the button that creates the object. (e.g., Add Signal, Add Clock, Merge Bus, Delay, Setup, or Hold buttons). This will open up the *Modify Auto Name Prefix* dialog box.
- Type in the new prefix and press the **OK** button.


Modify Auto Delay Name Prefix
Enter a new auto delay name prefix:
Delay
OK
Cancel

Chapter 2: Clocks

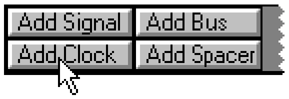
Clocks are special repetitive signals that draw themselves based on their attributes: period, frequency, duty cycle, edge jitter, offset, and other parameters. Clocks can be related to other clocks by using the Master Clock property or by using formulas that reference the attributes of another clock.

Clocks are a specialized form of a signal, so they are selected, moved, deleted, and hidden just like regular signals. However, clock edges are fixed and cannot be pushed by a delay parameter or dragged using the left mouse button.

2.1 Adding Clocks

To add a new clock:

- Left click on the **Add Clock** button to open *Edit Clock Parameters* dialog.



- Enter the values for the new clock.
- Click the **OK** button to close the dialog.

2.2 Edit Clock Parameters Dialog

To edit an existing clock:

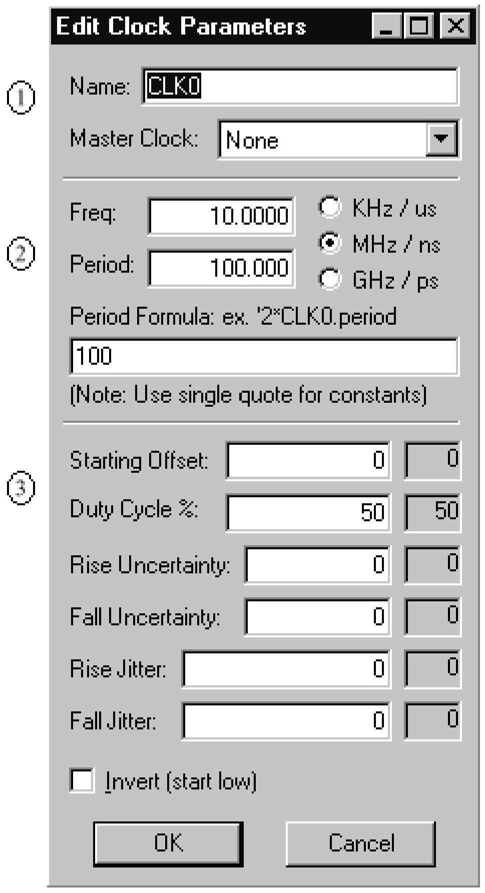
- Double left click on a segment in the clock waveform to open the *Edit Clock Parameters* dialog.

OR

- Double left click on the clock name in the label window to open the *Signal Properties* dialog. Click the **Clock Properties** button, in the center of the dialog, to open the *Edit Clock Parameters* dialog.

There are three input sections in the *Edit Clock Parameters* dialog: (See Figure 2.1 on the following page)

1) **Label Section:** There will be a valid default clock name already in the edit box. User generated names must be different from any other clock, signal, or bus name. The names must be at least one character (no spaces are allowed). Place the suffix $BAR at the end of a name to indicate the signal is active low. Active low signal names are displayed with a bar (line) drawn over the name.



Edit Clock Parameters

Name: CLK0

Master Clock: None

Freq: 10.0000

Period: 100.000

KHz / us [✓ MHz / ns] GHz / ps

Period Formula: ex. '2*CLK0.period

100

(Note: Use single quote for constants)

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

☐ Invert (start low)

OK Cancel

Figure 2.1 Edit Clock Parameters Dialog

Master Clock lets you relate the current clock to another clock in the diagram. The next section *Master Clocks and Using Formulas in Clocks* covers this feature.

2) **Clock Rate Section:** The clock rate can be entered as either a *frequency* or a *period* (the other will adjust itself). These must be positive real numbers whose units are controlled by the radio buttons to the right of the edit boxes. If you encounter clock frequency rounding errors then lower the Base Time Unit.

The clock rate can also be set using the *clock period formula* box, covered in section 2.3.

3) **Clock Properties Section:**

- Clocks are normally high at time zero. *Starting offset* shifts the starting edge transition forward by the time value entered. The offset edit box accepts a valid time value or time formula.

- *Duty cycle* determines how long the clock is high during a given period. Duty cycle is a percentage of the period (0 < duty < 100). The duty cycle edit box accepts a valid time or time formula.

- Edge *uncertainty* is the amount of time that the clock may or may not be valid during one of its transitions. Adding uncertainty to the clock edges causes them to be drawn with gray uncertainty regions after the occurrence of the edge. The uncertainty edit boxes accept valid time values or valid time formulas.

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

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

Note: Grid Lines for clocks are now set by double clicking on the clock name to open the *Signal properties* dialog. A later section *Gridlines on Clocks* covers this feature in more detail.

2.3 Master Clocks and Using Formulas in Clocks

In order to model circuits that modify the system clock (like a "divide by 2" circuit or a clock distribution chip), it is necessary for some clocks to be related to other clocks. Master clocks and clock formulas allow a clock to reference another clock's attributes, thereby relating clocks to each other.

The attributes that can be referenced by another clock are **period**, **duty**, **offset**, rising edge jitter **(jrise)** and falling edge jitter **(jfall)**. To reference an attribute, use the dot method used in all time formulas: **CLOCK_NAME.ATTRIBUTE**. For example **CLK1.period** would retrieve the value of CLK1's period and **CLK2.jrise** would get the time value of CLK2's rising edge jitter.

The Master Clock field in the *Edit Clock Properties* dialog provides quick method of copying all the attributes of one clock into the current clock. After a master clock is selected, each attribute can be determined by a formula based on a master clock attribute.

The duty cycle, offset, rising, and falling edge jitter edit boxes in the *Edit Clock Parameters* dialog can take a time value or a time formula. To use a formula to describe the period, the **period formula** edit box must be used. It is separate from the normal period box because all formulas are assumed to be in display time units, and the period box is in the units indicated by the radio buttons to the right of the box.

A valid time formula is any algebraic combination of clock_name-dot-attribute, free parameters, time values and constants indicated by a single quote. The following are examples of valid time formulas:

2*CLK0.period = the constant 2 times the period of CLK0

CLK1.period/2 = Divide the period of CLK1 by the constant 2

2*CLK0.period + F0.min + 5 = the constant 2 times the period of CLK0 + the minimum time value of free parameter F0 + 5 display time units

In the last formula, the 2 is implicitly assumed to be a constant because CLK0.period is a time value and time formulas must evaluate to units of time (not time squared). Similarly, the 5 is assumed to be a time value. For more information on formulas, read the sections on *Time Formulas* and *Operators* in Chapter 5.

2.4 Editing Clock Parameters

Most clock parameters are edited using the *Edit Clock Parameters* dialog covered in the earlier section. This is a fully interactive dialog that changes the clock as soon as you change a clock attribute. This makes it easy to immediately assess the impact of a different clock frequency on your design. To edit a clock:

- Double left click on a segment in the clock waveform to open the *Edit Clock Parameters* dialog.

However, some clock parameters which are common with signals are edited using the *Signal Properties* dialog. These parameters include grid lines, VHDL type, Verilog type, direction, export status, and grid lines. To edit a clock's signal properties:

- Double click on the clock name to open the *Signal Properties* dialog.

What happens when the period of a clock changes? If the period of the clock changes, parameters will stay attached to the same numbered transition. For example:

- A clock has a transition every 50ns and a parameter is attached to the 3rd transition at 150ns.
- Then the clock period is changed so that the transitions occur every 100ns.
- The parameter will still be attached to the third transition only now it is at 300ns, instead of 150ns where there is no longer a clock transition.

2.5 Inserting and Deleting Clock Cycles

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

To insert or delete cycles in a clock:

- Select a clock edge. For inserting, click on the clock edge immediately to the right of where you want the cycles to be inserted. For deleting, click on the first edge you want to delete.
- Choose either the **Edit > Insert Clock Cycles** or **Edit > Delete Clock Cycles** menu option. This will open the *Clock Alteration* dialog. These functions are active only when a clock edge is selected.
- Type in the number of clock cycles to perform the function on.
- Press the **OK** button to close the dialog.

2.6 Measuring Clock Transition Times

To determine the exact time of a clock transition:

- Double left click on a clock transition to open the *Edge Properties* dialog.
- The exact edge time is displayed in this dialog box. Push **Cancel** to close the dialog.

Clock transitions occur at fixed intervals so you cannot move an individual transition. To adjust the entire clock, double click on a clock segment to open the *Edit Clock Parameters* dialog.

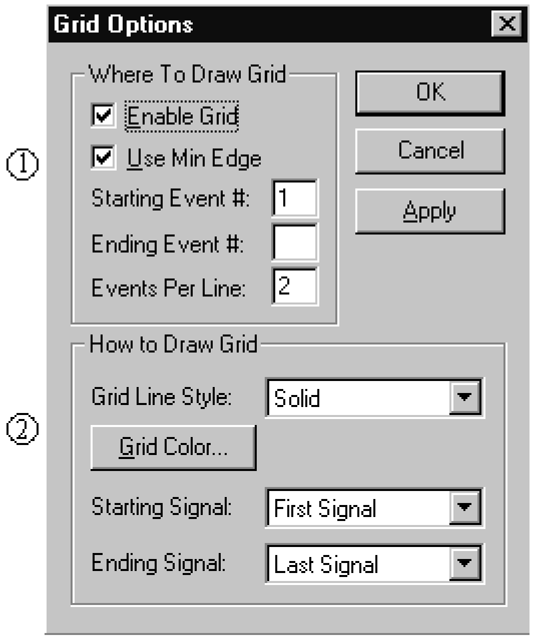
2.7 Grid Lines on Clocks

Grid lines are vertical lines drawn from the edges of a clock.

To draw a clock grid:

- Double click on the clock name to open the *Signal Properties* dialog.
- Press the **Grid Lines** button in the top right of the dialog. This opens the *Grid Options* dialog.
- Check the **Enable Grid** checkbox. This enables the rest of the grid options. If you later decide to turn off the grid then uncheck this check box.
- Fill in the rest of the controls to design a grid pattern. Use the **Apply** button to test your grid settings.

The *Grid Options* dialog has two sections (See Figure 2.2 below):



Where To Draw Grid

- [✓] Enable Grid
- [✓] Use Min Edge
- Starting Event #: 1
- Ending Event #:
- Events Per Line: 2

How to Draw Grid

- Grid Line Style: Solid
- Grid Color...
- Starting Signal: First Signal
- Ending Signal: Last Signal

Figure 2.2 Grid Options Dialog Box

1. Where to Draw Grid Section:

- The **Enable Grid** check box determines whether the grid lines are turned on or off.
- The **Use Min Edge** check box determines if grid lines are drawn from the minimum or maximum edge of a clock transition. This affect takes place when edge jitter or uncertainty has been selected.
- The **Starting Event #** accepts a positive integer representing the event number where the first grid line is to be drawn. The first event in the clock is the 0 event.
- The **Ending Event #** accepts a positive integer representing the event number where the last grid line is to be drawn.
- The **Events Per Line** edit box determines how many clock transitions occur before another grid line is drawn. The events per line edit box accepts positive integers, where 1 draws a grid line on every transition, and 2 skips 1 transition between grid lines, etc.

2. How to Draw Grid Section:

- The **Grid Line Style** drop down box determines what kind of grid line is drawn. All the standard Windows line styles are available: solid, dash, dot, dash-dot, and dash-dot-dot.
- The **Grid Color** button allows the user to set the color of the grid lines.

- The **Starting Signal** and **Ending Signal** boxes determine the length of the grid lines. The default values cause the gridlines to be drawn from the top of the diagram to the bottom.

Chapter 3: Group and Virtual Buses

A bus is a signal that displays the values of a group of member signals. Sometimes it is important to be able to uncouple the bus and make use of a value on an individual member signal. At other times, the bus will always be treated as a whole (no individual signal values are needed). In order to take advantage of this distinction there are two kinds of buses: **Group Buses** for working with individual member signals, and **Virtual Buses** for quickly and easily displaying a signal that looks like a bus.

- **Group Buses** are composite signals whose transitions and state values are determined by their member signals. Instead of individually editing related signals (like the address lines of a part), a bus can compress all the signals' data into one compact signal. The individual bus signals can be uncoupled, or displayed along with the bus. Buses are added using the **Add Bus** button and selecting a bus type of **Group Bus**.

- **Virtual Buses** are normal signals that are drawn with Valid, Invalid, and TriState states and use *virtual state information* to represent bus values. Virtual buses are added using the **Add Signal** button or using the **Add Bus** button and selecting a bus type of **Virtual Bus**. The state information is added using the **HEX** state button. A virtual bus doesn't have member signals.

Virtual buses are faster and easier to work with than group buses, because group buses cause many signals to be generated. For instance, a 32 bit group bus means that 33 signals (32 member signals and one bus signal) are drawn and updated each time the bus changes. If a delay parameter ends on a bus edge all the member signals must be updated when the delay is changed. Use group buses only when you need to get access to an individual member signal at some point in your design. Virtual Buses only need the same memory resources as any other individual signal, regardless of the size that they represent (e.g., 8,16, 32 or more bit bus). Group buses are primarily useful when you have several signals already existing (possibly imported from a simulation run or from a logic analyzer) and you want to view them as a bus value.

Virtual Buses are the recommended way to display and work with bus information. When exporting to VHDL and Verilog stimulus files and test benches, Virtual Buses are exported as vectors. To learn more about Virtual Buses read:

3.1 Adding and Editing Virtual buses

3.2 Virtual State Information

Group Buses should only be used when the member signals need to be separated and used outside the bus. For instance if you have imported eight address signals from a simulator and would like to view the value of the address bus, then these signals can be easily compressed into a group bus and displayed as one signal. When exporting to VHDL or Verilog, only the member signals are exported (there is no vector that represents the bus). To learn more about group buses read:

3.3 Adding Group Buses

3.1 Adding and Editing Virtual Buses

If you don't need to display the individual member signals of a bus, you can use the extended state information of a single signal to make a virtual bus. This will increase computational performance for timing diagrams that use large buses (32 bits or more) if timing parameters are attached to the buses.

To add a virtual bus:

- Click the **Add Signal** button to add a new signal.

- Sketch the bus waveform.

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

To set the width of a virtual bus:

- Double click on the name of the virtual bus to open the *Signal Properties* dialog.

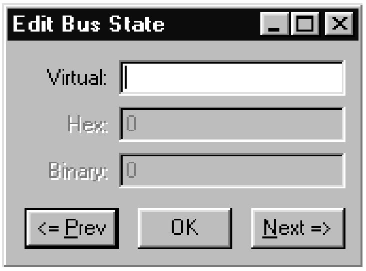
- Enter the MSB and LSB in the edit boxes at the bottom of the dialog, OR edit the signal name to include the MSB and LSB inside brackets (for example: **SIG2[31:0]** is a 32-bit bus).

Note: The MSB, LSB, and brackets are not part of the signal name. When using the signal in equations, use only the original signal name (for example: "SIG1 and SIG2[31:0]" is **incorrect** and will produce an error).

To edit the segment values of a virtual bus:

- Double left click on a segment to open the *Edit Bus State* dialog.

- Type the bus segment data into the **Virtual** field. Use the **<Alt>-N** or **<Alt>-P** keys or the **Next** and **Prev** buttons to move between the different segments on the virtual bus. Then click **OK** to close the dialog.



In the next section you will learn how to use the **Virtual** field to export user defined type information for VHDL files.

3.2 Virtual State Information

There are seven standard graphical states (high, low, tristate, valid, invalid, weak high, and weak low) which cover most circuit applications. However, some simulation languages require more complex state types such as integers and enumerated data types. To support these types, SynaptiCAD products allow virtual state information to be attached to signal segments. Any arbitrary text string can be used as a virtual state (e.g. A0C, 5 + 3, blue level, and 24 are all valid virtual states). Note that spaces are valid characters in a virtual state string.

Virtual state information is displayed in the center of a signal segment (the text is hidden if the segment is smaller than the text of the virtual state). Since a bus displays the states of its member signals in the same manner, virtual states are used to create virtual buses.

To add virtual state information to an existing signal:

- Double click on a segment in the signal. This opens the *Edit Bus State* dialog.
- Type any string into the **Virtual** field to set the virtual state information which will be displayed in the center of the segment.
- Click the **Next** or **Previous** button to edit the next or previous segment in the signal. Alternatively, use the **<Alt>-N** or **<Alt>-P** keyboard shortcuts (recommended for faster editing).
- Click **OK** to close the dialog box.

Note: The virtual state information associated with a transition is supplemental to the standard state assigned to a signal transition. The standard state (High, Low, Tristate, etc.) controls the graphical appearance of the segment and the virtual state sets the text in the segment. SynaptiCAD products do not attempt to interpret virtual state information. They only display and save the information. Waveperl scripts have access to virtual state information, and it is the script writer's job to define the meaning of virtual state information within a particular script. The VHDL and Verilog scripts use this information. See *Chapter 14: Writing Waveperl Scripts* for more information

3.3 Adding Group Buses

Group buses are composite signals whose transitions and state values are determined by their member signals. Instead of individually editing related signals (like the address lines of a part), a bus can compress all the signals' data into one compact signal.

Note: Buses are currently limited to a maximum size of 64 signals, but this will be increased in future versions.

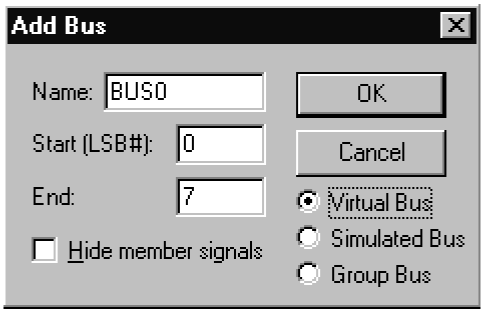
Before a group bus can be created, its member signal names must be selected or new signals need to be created.

To create a bus from existing signals:

- Select the signals in order from lowest bit to highest bit of the bus. Select signals by left clicking on the signal names.
- Left click on the **Add Bus** button or choose the **Bus > Add Bus** menu option.

To create a new bus and its member signals:

- Make sure that no signals are selected (clear selected signals by clicking in the Diagram window).
- Left mouse click on the **Add Bus** button on the left-hand side of the button bar or choose the **Bus > Add Bus** menu option. This will open the *Add Bus* dialog.



Name: BUS0
Start (LSB#): 0
End: 7
☐ Hide member signals
[✓ Virtual Bus] Simulated Bus Group Bus

- Type the name of the bus into the **Name** box. The member signals will be named the same name as the bus plus their signal number.
- Select the **Group Bus** radio button.
- Type in the **starting number** (least significant bit) of the bus. This will probably be either a 0 or a 1 depending on how you like to start counting.
- Type in the **ending number** (most significant bit) of the bus.
- Press the **OK** button. The signals including and in between the start and end numbers will be added and a bus will be created.

Note: Member signals are normally still visible after being merged into a bus. To change the default so that the member signals are hidden, use the **Options > Drawing Preferences** menu option and check the **Hide signals on bus merge** option in the dialog box. The **View > Show Hidden Signals** menu can be used to unhide the signals.

3.4 Editing Group Buses

Group buses are a special form of signal so they can be edited just like regular signals. Any changes made to the bus are reflected in the member signals. The following techniques can be applied to group buses:

Change the Bus Display: A group bus can display its data in hex or binary. To change from one representation to another:

- Choose the **Options > Drawing Preferences** menu option. This will open a dialog of the same name.

- Check the **Display Bus in Hex {else Bin}** check box to display the bus values in hexadecimal, or uncheck it to display the values in binary.

HEX Button Editing: By using the *Edit Bus State* dialog box, a bus can be edited by typing the value of a bus segment in hexadecimal or binary. To edit a bus segment with the *Edit Bus State* dialog:

- Double click on a segment to open the *Edit Bus State* dialog.
- The **Hex** box accepts numerals 0-F. The **Binary** box accepts numerals 0 and 1.
- Both boxes also accept L, H, Z, V, and X. Where L= weak low, H = weak high, Z = tristate, V = valid, and X = invalid state.
- Type in a new bus state into the Hex or Binary edit boxes and press the **OK** button to save the changes.

Normal Signal editing techniques:

- Drag and drop bus transitions by left clicking.
- Double left click on a bus transition to open up the *Edge Properties* dialog. With this dialog you can lock the bus transitions to a specific time or type in an exact location that the transition should occur.
- Left click on a segment and then left click on a STATE button to change all the signals in that segment to the new state (like all ones or zeros).
- Left click and hold inside of a bus segment, and drag to insert a new bus state.

3.5 Binding and Unbinding Group Bus Edges

Binding a group bus edge adds invisible links to all the member signal transitions that occur at the same time as the bus edge. This means that when a bounded member signal is moved, the bus transition will also move.

To Bind a bus edge:

- Left click on the bus edge to select it.
- Choose the **Bus > Bind Bus Edge** menu option. The edge will now be bound. To test the binding drag a member signal edge and see if the other signal edges move.

To Unbind a bus edge:

- Left click on the bus edge to select it.
- Choose the **Bus > Unbind Bus Edge** menu option. To test the unbinding drag a member signal edge and see if the other signal edges do not move.

Note: Bus edges are automatically bound when a delay is added to the bus transition.

Tip: If you wish to bind edges and they are not at the exact same time, use the **align bus edge** command first before binding.

3.6 Aligning Group Bus Edges

The **Bus > Align to Bus Edge** menu function moves all the nearby member signal transitions to the exact time of the selected bus edge. Since each bus edge is caused by a member signal edge, misaligned signal edges cause several bus edges to occur close together.

To align all member signal transitions within a ten pixel range:

- Left click on the bus edge to select it.
- Choose the **Bus > Align to Bus Edge** menu option. This will cause all the signal transitions within ten pixels of the bus edge to move to the exact time of the bus transition (if the bus edge is a max edge, signal transitions are aligned using their max time).

Note: Align edges before adding a delay to a bus transition. When a delay is added to a bus edge all member signal transitions that occur at that specific time are *bound* and will move with the bus delay. If any member signal transitions are not aligned, they will not be affected by the delay.

3.7 Expanding Group Buses

Expanding a bus deletes the bus and any parameters or text attached directly to the bus. The member signals of the bus are automatically shown if they were hidden. To expand or delete a bus:

- Left click on the bus name, then choose the **Bus > Expand & Delete Bus** menu option.

Note: Buses can be deleted by selecting the bus name and pressing the delete key. This operation does not affect the member signals, so hidden member signals will stay hidden.

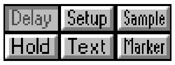
Chapter 4: Parameters - Delay, Hold, Setup, and Sample

Parameters actively move and monitor signal transitions. They provide the automatic updates and timing analysis capabilities of the timing diagram editor. There are four types of parameters:

- *Delays* specify a fixed time between two signal transitions.
- *Setups* and *Holds* specify a minimum time that data must be stable with respect to a control signal.
- *Samples* indicate a point at which a signal is to be "sampled" for its value (for example, sampled by a latch or register). Samples are used by TestBencher Pro to specify how to generate the self-testing code in a test bench.

To add a timing parameter:

- Press on one of the **Delay**, **Setup**, **Hold**, or **Sample** buttons to turn the button red.

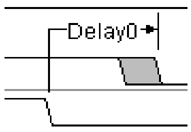


- Left click on a transition to select it. For a delay this is the forcing edge. For a setup or hold this is the transition that will be monitored.
- Right click on the second transition to add a parameter between the first and second transitions. For a delay this is the transition that will be moved. For a setup or hold this is the control signal.

When a parameter is added to a timing diagram it is blank and does not perform any timing analysis calculations until you edit the parameter properties.

4.1 Delays

A delay specifies a fixed time between two signal transitions. Both the **min** and **max** values are used in defining delays. If the min and max values are different, the delayed signal transition will be displayed with a gray region of uncertainty. The **margin** parameter has no meaning for delays and is marked as "na" for not applicable in the parameter table.

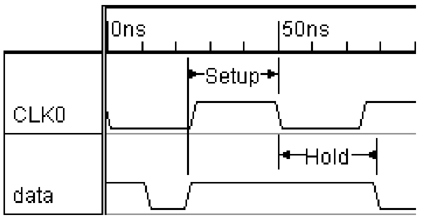


Visual Display of critical paths: Delays are color coded to indicate which edges of a transition they control. A **Gray** delay sets neither edge of the delayed transition, either because the min/max values are blank or another delay is dominant at the delayed transition. **Black** delays set both the min and max edges of a signal transition. **Blue** and **Green** indicate min only and max only edge setting. The delay color coding visually displays the critical paths through the diagram. This is especially useful when multiple delays end on the same transition. You can turn off the color coding for documentation purposes using the **View > Show Critical Paths** menu option.

Note: Delays cannot have circular dependencies - if A delays B, and B delays C, then C cannot delay A.

4.2 Setups and Holds

Setups and Holds check timing constraint requirements for a design. **Setups** are the minimum time necessary for a signal to be stable before a control signal transition. **Holds** are the minimum time that a signal must be stable after a control signal transition.



When drawing a setup or hold the first edge selected is the *data signal* and the second edge is the control signal. To remember this order, remember that data signals have setup/hold constraints whereas control signals like clocks do not.

Note: *If drawn correctly, the arrows will point toward the control signal.* If drawn backwards the two arrows will point toward each other.

Setups and Holds have three values of interest:

- The **min** for a setup is the minimum time that the data signal transition can occur before the control signal. The min for a hold is the minimum time that the data signal transition can occur after the control signal.
- The **max** column is optional for constraints and is rarely used. If a max constraint is specified, then the signal transition has to occur between the min and max times.
- The **margin value** (slack) is calculated by the program. It indicates the amount of safety margin available before the constraint condition is violated. When a constraint is violated, it will turn red in both the drawing and the parameter windows. The parameter can also be made to display the margin value by setting the **Display Label** to **min/max Margin** in the *Parameters Properties* dialog box.

4.3 Samples

Samples are graphical constructs which indicate a point at which a signal is to be "sampled" for its value (e.g., sampled by a latch or register). Samples do not have any computational effects on timing or the states of signals. Their primary use in Timing Diagrammer Pro and WaveFormer Pro is for documentation. In TestBencher Pro these samples indicate how to generate the self-testing code for the test bench.

Samples can either placed at a fixed point in time or they can be relative to a triggering edge transition. The fixed or relative triggering is determined by the way you draw the sample.

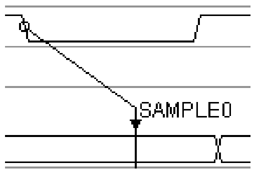
Both **min** and **max** time values are used in defining samples. If the min and max values are different, the sample will be displayed as a boxed time range on the sampled signal (a "sample window"). On absolute samples, the min/max times represent the absolute sample time. On edge relative samples, the min/max times represent the sample time relative to the sample's trigger edge. The **margin** parameter is not used in samples.

To add a sample:

- Press the **Sample** button to turn the button red.
- Decide if you would like to attach the sample to a fixed time or to an edge.

  If a sample is attached to a fixed time, it will stay at that time. If a sample is attached to an edge, the sample will sample at a fixed distance from that edge.

- If attaching to a time, make sure a signal edge is not selected by selecting something else like a parameter or a signal name.
- If attaching to an edge, select that edge by left clicking on it.
- Place mouse cursor on the signal to be sampled at the point in time at which you want to sample it's value.
- Right click to place the sample.:



To edit the values of a sample

- Double left click on the sample to open the *Parameter Properties* dialog box. The min and max edit boxes accept time values or time formulas.

4.4 Parameter Properties

Use the *Parameter Properties* dialog to edit a parameter. To open the *Parameter Properties* dialog (see Figure 4.1 on the following page) do one of the following actions:

1) Double left click on the parameter in the Diagram window (to edit a single parameter instance),

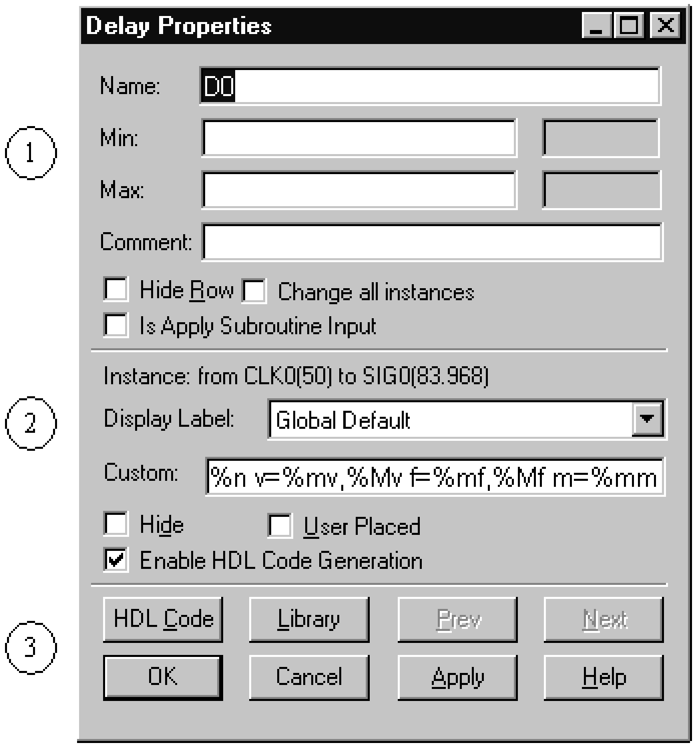
2) Double left click on the parameter in the parameter table window (to edit all parameter instances), or

3) Single left click on the parameter in the parameter table and start typing on the parameter in the parameter table and the dialog will open.

Note: Double clicking on a parameter when the dialog is already open causes the dialog to begin editing the double clicked parameter. This is a handy technique for quickly editing multiple parameters.

1) Data common to all instances of a parameter:

- **Name:** There will be a valid default name already in the edit box. User generated names must be different from any other parameter name. The names must be at least one character long (no spaces are allowed). Parameter names must not contain any mathematical operators (+, -, ',*, /) and they cannot be a legal time value (like 5 or 7404). Parameter names beginning with "$$" are reserved for internal use.
- The **Min** and **Max** edit boxes accept time values or time formulae like **(D0.min* '2 + 10)**. Chapter 5 contains detailed information on time formulas and variables.


Delay Properties

Name: D0
Min:
Max:
Comment:
☐ Hide Row ☐ Change all instances
☐ Is Apply Subroutine Input

Instance: from CLK0(50) to SIG0(83.968)
Display Label: Global Default
Custom: %n v=%mv,%Mv f=%mf,%Mf m=%mm
☐ Hide ☐ User Placed
☑ Enable HDL Code Generation

Figure 4.1 Parameter Properties Dialog Box

- The **Comment** cells accept any string of characters, numbers, and spaces.
- The **Hide Row** check box hides the parameter in the *Parameter* Window.
- The **Outward Arrows** checkbox only appears in the Setup and Hold Parameter Properties dialogs. This control switches the parameter from the traditional databook format of “outward pointing arrows” to the default display of "point to control signal" style arrows. Outward arrows are useful when using setups as distance indicators or pulse width checks.
- **Change all instances**: if checked, the dialog controls in the second section (described below) affect all instances of the parameter in the Diagram window. For example, if there are two parameters in the Diagram window called D0, they will share the timing information displayed in the Parameter window. However, each instance of D0 in the Diagram window can be displayed differently. If you are editing a particular instance in the Diagram window then that instance will be selected so you can tell which one you are editing.

- The **Is Apply Subroutine Input** checkbox is a TestBencher Pro feature. This option allows certain parameter settings to be variable inputs for a timing transaction.

- **Realistic Data Book Documentation:** The **Name**, **Min**, **Max**, and **Comment** edit boxes can include text that is subscript, superscript, bold, or italic. Select the text then use the following keys to toggle the text formatting: bold = **<Ctrl>-b**, italicize = <**Ctrl>-i**, superscript **<Ctrl>-u** (u for up), subscript = **<Ctrl>-d** keys (d for down).

2) Data individual to each parameter instance

- **Display Label** drop down list box: determines how a parameter displays itself in the Diagram window. The **Global Default** will display the setting checked in the *Drawing Preferences* dialog box. **Name, min/max Value, min/max Formula, min/max Margin**, and **Comment** will display the named attribute as it appears in the parameter window. **Distance** displays the minimum and maximum distances between the transitions (this value takes into account reconvergent fanout effects). **Custom** is a user defined string that is contained in the custom edit box. If you are changing all instances of a parameter and the individual instances have different display labels then a **Varies** in the Display Label edit box will indicate this condition.

- **Custom:** displays a custom string of characters and attributes. Certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a **%** character followed by one or two letters. The default equation contains one of each of the control codes, so if you start by editing this equation you do not have to memorize the control codes. The control codes are:

- name = **%n**		
- min value = **%mv**	and	max value = **%Mv**
- min formula = **%mf**	and	max formula = **%Mf**
- min margin = **%mm**	and	max margin = **%Mm**
- min distance = **%md**	and	max distance = **%Md**
- comment = **%c**		

- **Hide** check box hides a parameter instance in the Diagram window.

- A parameter can be fixed to a specific vertical position by checking the **User Placed** checkbox.

- The **Enable HDL Code Generation** checkbox is a TestBencher Pro feature. It allows the HDL code generation for a specific parameter to be enabled and disabled.

3) Button section:

- **HDL Code** button opens a dialog that is used to generate the self-testing code for TestBencher Pro samples.

- **Library** button opens the *Library Parameters* dialog so that you can select a library parameter to reference in your min and max formulas.

- **Prev** and **Next** buttons cycle through all the parameters in the diagram so that you can quickly edit multiple parameters.
- The **Apply** button allows you to preview your changes before you change the parameter permanently.
- The **Cancel** button restores the original values, and the **OK** button saves the changes.

4.5 Deleting Parameters in the Diagram window

To delete a parameter from the Diagram window:

- Select the parameter by left clicking on it. A selected parameter has a selection box drawn around it.
- Do one of the following:

  a) Press **<Delete>** on the keyboard.

  b) Choose the **Edit > Delete** menu option.

  c) Use the right mouse button in the Delete Mode. To enter the delete mode choose the **Edit > Right Click Delete Mode** menu option.

4.6 Smart Parameter Sharing (reusing parameter values)

The program assumes that multiple parameters between the same signals and the same types of edge pairs represent the same timing. Therefore the parameters will share the same name and timing values. This is called smart parameter sharing.

Example: Two signals, A and B, are made up of high and low segments. Four different delays can be added before smart sharing will start sharing the data (Ahl->Bhl, Ahl->Blh, Alh->Bhl, Alh->Blh). If another parameter is added then that parameter will have the same name as the parameter that is between the same types of edges.

Overriding a smart sharing (on and off)

If a parameter is shared or not shared when you wanted otherwise, change the name of the parameter in the Diagram window to correct this situation. To force or undo parameter sharing:

- Double left click on the parameter name to open the *Parameter Properties* dialog.
- If you want to **force** a parameter sharing, then type in the name of an existing parameter.
- If you want to **undo** a parameter sharing, then type in a unique parameter name.

Parameters cannot be shared by changing the name of the parameter in the parameter window.

Turning off smart sharing

To turn off smart sharing so that parameter data is not automatically shared:

- Choose the **Options > Design Preferences** menu option. This will open up a dialog of the same name.
- Uncheck the **Smart parameter sharing ON** check box.
- Choose **OK** button to close the dialog.

4.7 Smart Parameter Lookup (reusing parameter values)

When a parameter name is edited inside the *Parameter Properties* dialog box, the name is automatically checked against the names of all the free parameters and library parameters that are available to the project. If a match is found with a free parameter then the parameter will use the free parameter's data and the free parameter will be converted to match the type of the parameter. If a match is found with a library parameter then the parameter will use the library parameter's data.

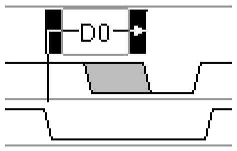
To use Parameter Lookup:

- Double left click on the parameter in the Diagram window. This will open the *Parameter Properties* dialog box.
- Change the name of the parameter to match the name of a free parameter or library parameter. Capitalization and any library specifications must match exactly.
- Choose **OK** to exit the *Parameter Properties* dialog. This will cause a message box warning that you are about to share data.
- Choose **OK** to close the message box.

4.8 Moving Parameter Start and End Positions

Drag and drop the solid handle boxes on either side of the selected parameter to change a parameter's starting or ending signal transition. To do this:

- Left click on the parameter to select it. A selected parameter is surrounded by a rectangle with a solid handle box on either end.
- Place the mouse over the handle box nearest the transition that is being changed.
- Left click and drag the mouse so that the new edge is highlighted. If the entire parameter is changing its **vertical position**, then you clicked on the middle of the parameter instead of a handle box.
- Release the mouse button to attach the new edge.



Warning: Moving the start and end positions of delays can radically change the way your timing diagrams look, because delays force transitions to be a specific distance apart. You can use the **Edit > Undo** menu option to return the timing diagram back to its original state.

Note: To abort the selection of a new edge during the process, press the **<Esc>** key.

4.9 Hiding and Showing Hidden Parameters

Hiding a parameter removes that parameter from the screen, but not from the project. Delays, setups, and holds will continue to function, but timing errors will be shown only in the parameter window.

There are two methods that can be used to hide parameters. The first method works by setting the hide flag of selected parameters. The second method works by using filter patterns to match against the parameter names (this method is particularly useful if your project has hundreds of parameters). Each hiding method operates independently so both methods can be used to hide different parameters at the same time.

1) To hide or show parameters using the hide flag do one of the following:

- Select the parameter in the Diagram window, then choose the **View > Hide Selected Parameter** menu option.
- OR, double click on the parameter in the Diagram window to open the *Parameter Properties* dialog, then check the **Hide** check box and apply that change.
- Use the **View > Show Hidden Parameter** menu option to show a parameters that were hidden using this method. This menu brings up a dialog box that allows you to selectively unhide parameters. If a parameter is attached to a hidden signal, it will continue to stay hidden until the signal is shown.

2) To hide or show signals using filter patterns:

- Choose the **View > Filter Parameters** menu option to open the *Parameter Filter* dialog. This is a modeless dialog that can be left open while you work with the diagram.
- Choose either the **Show Only** or the **Hide** radio button to determine how the filter patterns act. **Note:** if at some point you wish to see all the parameters that are hidden by the patterns, choose the opposite radio button.
- Enter a filter pattern in to the **Pattern** edit box. For example, the pattern **D** would match parameters named **D0** and **D1**.
- Press the **Add** button to activate the pattern. Multiple patterns can be entered.
- To delete a pattern, select it and press the **Delete** button.

Chapter 5: Timing Analysis and Time Formulas

This chapter describes the advanced timing analysis and time formulas supported by the timing diagram editor.

Timing Analysis Functions: The timing analysis section describes a) how to control the result of multiple delays driving a single edge, b) reconvergent fanout, and c) advanced properties of delays and signals.

Timing Formula Functions: Time formulas can generally be used wherever a time value is required (e.g., in the edit boxes of parameter min, parameter max, clock period, clock offset, and clock jitter). Time formulas allow parameters and clocks to reference each other. Clocks can be related to each other to model clock distribution circuits. Parameters can be related to each other by library values or by formulas based off a common gate delay.

This chapter also describes how to write time formulas, the special dot-attributes properties that let you access specific values, variables and free parameters, and macros.

5.1 Effects of Multiple Delays Forcing the Same Transition

If more than one delay ends on the same transition, then it is not clear when the forced transition should occur. There are four different ways that this ambiguity can be resolved:

- **earliest transition:** The earliest min edge and the earliest max edge will determine the uncertainty region of the delayed edge. Use this method when a transition occurs after one of several transitions has occurred.
- **latest transition:** The latest min edge and the latest max edge will determine the uncertainty region of the delayed edge. Use this method when a transition occurs only when all of a set of transitions have occurred.
- **max uncertainty:** The earliest min edge and the latest max edge will determine the uncertainty region of the delayed edge (maximizes the uncertainty from forcing edges).
- **min uncertainty:** The latest min edge and earliest max edge will determine the uncertainty region of the delayed edge (typically not very useful).

Global Default Setting: To set the default method which transitions will use to reconcile multiple delays:

- Choose the **Options > Design Preferences** menu option. This will open a dialog of the same name.
- Choose the specific method under the **Reconciling Multiple Delays Default** section.

Individual Setting: Normally transitions use the default method to reconcile multiple delays. However each transition can select a specific method by using the *Edge Properties* dialog. To set the method for a specific transition:

- Double click on the transition to open up the *Edge Properties* dialog.

- Choose a specific method under the **Multiple delay resolution** section.

Min only or Max only Delays: Normally delays have both a min and a max value even if the values are the same. However if you specifically clear either the min or max value then the delay becomes a min only or max only delay. A min only delay just affects the min edge of a transition and a max only delay just affects the max edge of a transition. If a transition has only min or only max only delays attached to it, the unconstrained edge (e.g., the max edge if min only delays are attached) will be set to the same value as the constrained edge.

Delays are color coded to indicate which edges of a transition that they control. A **Gray** delay does not set either edge of the delayed transition, either because the min/max values are blank or another delay is dominant at the delayed transition. **Black** delays set both the min and max edges of a signal transition. **Blue** and **Green** indicate a min only and max only edge setting respectively. The delay color coding visually displays the critical paths through the diagram. This is especially useful when multiple delays end on the same transition. You can turn off the color coding using the **View > Show Critical Paths** menu option.

5.2 Edge Properties Dialog

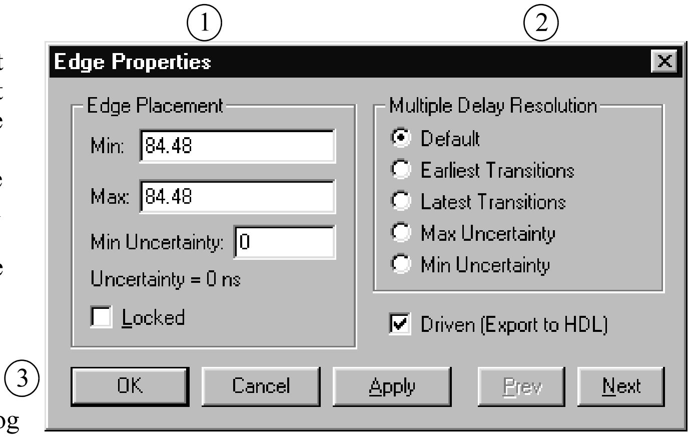
The *Edge Properties* dialog box displays information about a specific signal transition.

To open the *Edge Properties* dialog box:

- Double left click on either edge of the signal transition.

There are three sections in the *Edge Properties* dialog (see Figure 5.1 below):

1. Edge Placement Section: This section contains **Min** and **Max** edit boxes which display the placement of each edge of the transition. If the transition is not locked, then the edge can be placed at an exact time by typing time values into the min or max edit box. You cannot add uncertainty to the edge by using the min and max edit boxes.



Min: 84.48
Max: 84.48
Min Uncertainty: 0
Uncertainty = 0 ns
☐ Locked
[✓ Default] Earliest Transitions / Latest Transitions / Max Uncertainty / Min Uncertainty
☑ Driven (Export to HDL)

Figure 5.1 Edge Properties dialog

The **Min Uncertainty** edit box lets you add a minimum uncertainty region to a signal transition. If no delays cause an uncertainty greater than the transition's minimum uncertainty, the transition will be given its minimum uncertainty value.

The **Uncertainty** = X**ns** displays the actual width of the uncertainty region.

If checked, the **Locked** checkbox fixes a transition at a specific time. This also indirectly locks any transitions connected to the locked transition by a defined timing path. Clock transitions are always locked.

2. Multiple Delay Resolution Section: This section controls where a transition is placed when multiple delays end on it. These settings only affect edges that have more then one delay forcing their edge placement (see section 5.1 Effects of Multiple Delays Forcing the Same Transition).

3. Button Section:

- The **Apply** button allows you to preview your change before you actually change the transition time.
- **Next** and **Prev** buttons, or **<Alt>-N** and **<Alt>-P** keystrokes, simplify editing of multiple edges by letting you move to different edges on the same signal without closing the dialog.
- The **Cancel** button restores the original values, and the **OK** button saves the changes.

5.3 Effects of Common Delays (Reconvergent Fanout)

When two **timing paths** share a common transition, the uncertainty of that transition should not affect distance and margin constraints between transitions further down on the timing paths. This is because the common transition occurs at the same time for both timing paths. This effect is called **reconvergent fanout** because it usually occurs when a signal fans out to multiple gates and the outputs of these gates reconverge as inputs to a common gate. If the effects of reconvergent fanout were ignored, a design may appear to violate a timing parameter, even when the design would work because the margin calculations would be overly pessimistic.

The timing diagram editor automatically accounts for reconvergent fanout effects, correctly calculating margins and distances for parameters. Reconvergent fanout does not change the uncertainties of individual transitions so they are not visible on the timing diagram itself (except for constraint margins and distance values).

Example: A NAND gate has an uncertainty region of 10ns and a fanout of and signal paths that go through some gates and reconverge at the data and clock inputs of a D Flip-Flop. The difference in arrival times of the two paths should just be the delay and uncertainties of the two paths. The 10ns uncertainty of the NAND gate should not effect the difference in arrival times, because both signals will start at the same time (somewhere in the uncertainty region of the NAND gate).

See Figure 5.2 on the following page for an example of reconvergent fanout.

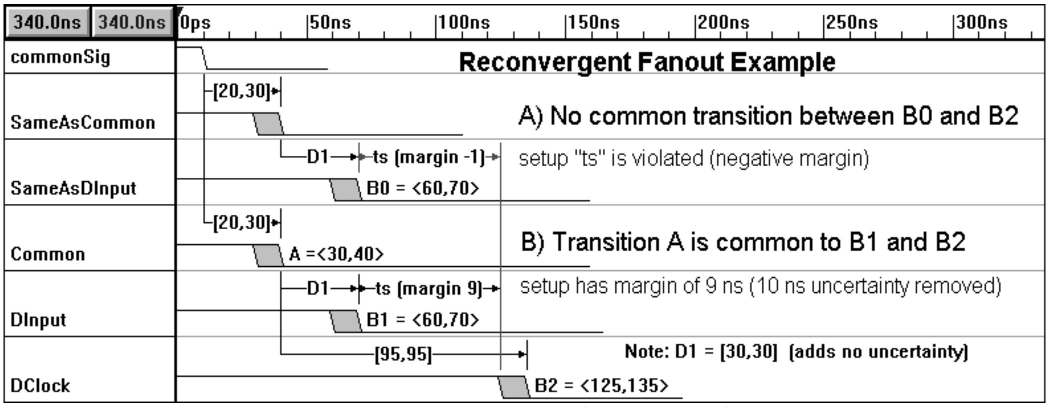

340.0ns
340.0ns
0ps
50ns
100ns
150ns
200ns
250ns
300ns
commonSig
SameAsCommon
SameAsDInput
Common
DInput
DClock
Reconvergent Fanout Example
[20,30]
A) No common transition between B0 and B2
D1
ts (margin -1)
setup "ts" is violated (negative margin)
B0 = <60,70>
[20,30]
B) Transition A is common to B1 and B2
A =<30,40>
D1
ts (margin 9)
setup has margin of 9 ns (10 ns uncertainty removed)
B1 = <60,70>
[95,95]
Note: D1 = [30,30] (adds no uncertainty)
B2 = <125,135>

Figure 5.2 Reconvergent fanout example

5.4 Locking and Unlocking all edges on a Signal

All the edges on a signal can be locked or unlocked at the same time. Locked edges are fixed in time and cannot be dragged or forced to move by a delay. To lock or unlock all edges on a signal or signals:

- Left click on the signal names to select them.

- Choose one of the menu options

Edit > (Un)Lock Edges of Selected Signals > Lock *or* **Unlock**

Individual edges can be locked or unlocked using the *Edge Properties* dialog covered in the previous section.

5.5 Design Preferences Dialog

The *Design Preferences* dialog affects how functions work in the program. To open this dialog choose the **Options > Design Preferences** menu option in the timing window. The *Design Preferences* dialog controls the function of the following:

Middle Mouse Function: The middle mouse button either toggles between the last two activated **state buttons** or cycles through the state buttons activating each in turn. The default function is to "toggle state buttons".

Default Edge Setting for Multiple Delay Resolution: If more than one delay ends on the same transition, then it is not clear when the forced transition should occur. This is the global default method that transitions will use to resolve this ambiguity. See section 5.1 for definitions of the different methods. Default is **earliest transition.**

Note: Individual transitions can be made to use a different setting by using the *Edge Properties* dialog box.

Continuous measurement On: if checked, then the black and blue digital readouts continuously update as the mouse cursor moves. Uncheck this box if the flicker bothers you. Default is checked.

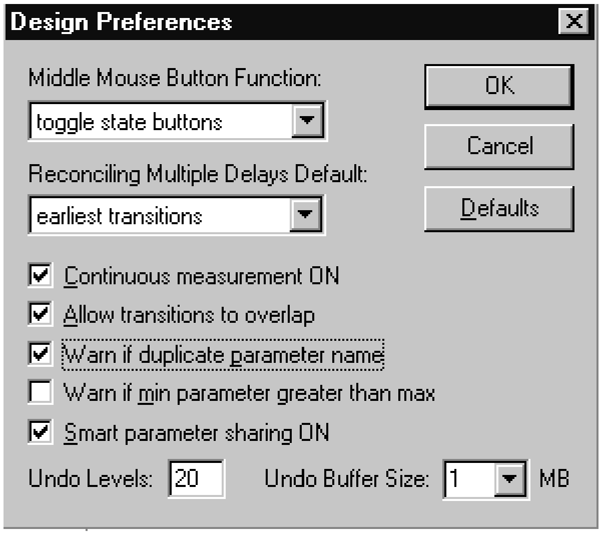
Allow transitions to overlap: Allows the transition regions of two events on the same signal to overlap. Default is checked.

Warn if duplicate parameter name: If a parameter name is edited in the Diagram window and it is changed to the same name as another parameter then the program assumes that you would like the edited parameter to share the same data as its new name implies. This could result in the deletion of the edited parameter's timing data. Normally the warning will be displayed.

Warn if min parameter greater than max: Normally the min value is less than the max value of a parameter. This is not necessary for proper functioning of parameters, and a warning can be annoying sometimes when several timing values are being edited. Normally the warning is displayed.

Smart parameter sharing ON: When multiple parameters are added between the same two signals and between the same edge types then it is assumed that the parameter represents the same timing value. This is called smart parameter sharing. Normally the function is ON.

Undo Levels: Sets the maximum number of drawing/editing actions that can be undone. Higher levels of Undo take up more memory



Middle Mouse Button Function: toggle state buttons

Reconciling Multiple Delays Default: earliest transitions

[✓] Continuous measurement ON
[✓] Allow transitions to overlap
[✓] Warn if duplicate parameter name
[ ] Warn if min parameter greater than max
[✓] Smart parameter sharing ON

Undo Levels: 20 Undo Buffer Size: 1 MB

Figure 5.3 Design Preferences Dialog Box

5.6 Time Formulas

A time formula is a mathematical expression for computing a time value. Time formulas can generally be used wherever a time value is required (e.g., in the edit boxes of parameter min, parameter max, clock period, clock offset, and clock jitter).

Time formulas are in display time units and are an algebraic combination of time values, constant values, clock properties, parameter names, and variables (free parameter names). Below are examples of valid time formulas:

- **Time Values** are fixed point numbers in display units. Fractional parts depend on base time unit and display time unit. Examples:

  5
  2.010
  255E-3
- **Constants** are fixed point numbers that are *unitless*. Multiplication and division operations automatically assume that the result is a value of time (not time squared) so constants are assumed to be where needed. The only time when a constant needs to be explicitly defined is when you plan to change the base time unit of an existing timing diagram. In this case, a *constant must be preceded by a single quote or they will be converted to display time units*.

  Constants do not scale (unlike time values) when the base time unit is changed. Example:

  '2
  '4.1e2
- **Parameter names**, used in formulas, reference their min and max time values according to the *operator rules* covered in the next section. The dot-max and dot-min properties can be used to force the equation to use a particular value. Clock properties can also be accessed using the dot attributes. Examples:

  D0
  delay7404
  delay7404.max
  Clock0.period
- **Algebraic combination** of the above types. Examples:

  5 + '2 * F0
  is parsed as 5(time value) + 2(constant) * F0(free parameter)
  D0.max - D1.min
  is parsed as D0.max(parameter D0's max value) - D1.min(parameter D1's min value)
  Clock0.period / '2
  is parsed as Clock0.period(Clock0's period value) / 2(constant)

Special Feature: Inside the *Parameter Properties* dialog, the **Library** button causes the *Library Parameters* dialog to open in the **View Parts** mode. This lets the user browse the library parts and use **Insert into Formula** button to copy a library parameter into the min and max edit boxes. (For more information read *Chapter 10: Using Library Parameters - Referencing.*)

Note: Time Formulas in parameters can not have circular dependencies. For example, assume A, B, and C are parameters. If A is referenced by B, and B is referenced by C, then C cannot be referenced by A. Each time a parameter is modified, the new value is checked for circular dependencies. When this error is detected, the new data will not be accepted and a message box will appear.

5.7 Operators and Dot Attributes

The timing diagram editor supports the **+, -, *,** and / mathematical operators. The dot attribute of parameters and clocks can override the default function of the mathematical operators.

Example: 4 ^ 2 = four squared

The dot attributes of parameters and clocks can override the default function of the mathematical operators.

Parameter Values: When a parameter name is used in a formula either the min or max value can be used during the evaluation process. By default the formula operator will use the min value if the formula is in the min column, and it will use the max value if the formula is in the max column. The user can force a particular value to be used by using the *parameter_name.min* or *parameter_name.max* attributes.

> **Example:** There are two free parameters, Afree = [10,20] and Bfree = [15,30]. Another parameter has the formulas [Bfree - Afree, Bfree-Afree] in the min and max columns. The timing diagram editor interprets this as [Bfree.min-Afree.min, Bfree.max-Afree.max] = [5,10].

There are a number of mathematical functions that the editor can use (see the online help for a complete listing). The function names are case sensitive and are all lower case. You do not need to use the parentheses to call a function, but it will be evaluated using the first number.

> **Example:** abs -1 + 1 = 2
>
> abs (-1 + 1) = 0

Clock Attributes: The properties of a clock that can be referenced in time formulas are: **period**, **duty**, **offset**, and rising **(jrise)** and falling **(jfall)** edge jitter. To reference a clock property use the dot-property method. For example:

CLK0.period

retrieves the value of CLK0's period.

CLK2.jrise

retrieves the value of CLK2's rising edge jitter.

5.8 Free Parameters

A Free Parameter is a special kind of parameter whose sole purpose is to be used as a variable in other parameters. They are called "Free" because these parameters are not attached to any signal transitions in the Diagram window. To add a free parameter:

- Press the **Add Free Parameter** button in the parameter window. A blank free parameter will be added to the parameter window.

Free Parameter Libraries: If your designs use the same kinds of components, then it is a good idea to save all the free parameters to a special library file that can be used by several projects. To save all the free parameters to a library file:

- Select the **Libraries > Save Free Parameters As Library** menu option.

To use a free parameter library file in a project:

- Select the **File > Merge Timing Diagram...** menu option.

Warning: If there are any duplicate free parameters when you merge, then the free parameter library file will overwrite the project's free parameters.

5.9 Using Macros in Formulas

Macros allow you to substitute one string or character for another in formula evaluation. Macros can be used for many purposes. You can use macros to reference library parameters with long names using smaller alternative names without having to alter the formal name used in the library. If you make macros for your library specifications and use the macros in your parameter formulas, you can analyze the impact of switching to parts from another logic family or another vendor by changing the macro values to other library specifiers (assumes parameter names remain the same across libraries.).

When using macro names in formulas, the name must be surrounded with percentage signs('%'). Any number of macros may be used in a single string. However, nesting of macros is not allowed.

To open the *Edit Macros* dialog:

- Select **Libraries > Macro Substitution List...** menu option to open the *Edit Macros* dialog.

The *Edit Macro* dialog is used to add, edit, and remove macros.

To add a macro:

- Type the macro name into the **Name** box. No percentage characters are allowed in macro names. The delimiting percentage characters are automatically added to the beginning and end of the name when the name is added to the list. You may type these characters if you wish but it is not necessary.

- Type the tab key or click your mouse in the **Value** box.
- Type the macro value (the string that you want in place of the macro name). **Note:** The value box is a drop-down list box that automatically displays all the library specifiers and values that you have defined before, so you do not have to re-type that information.
- Click the **Add** button to add the macro or type enter to add the macro and close the dialog box.

To edit an existing macro:

- Click on the macro in the macro list box. The name and value should appear in the edit boxes above the macro list.
- Edit the macro name and value in the appropriate edit box.
- Click the **Add** button to post the change to the macro list.

To remove a macro from the list:

- Click on the macro that you want to delete.
- Click the **Delete** button in the dialog.

Macro Example: Name => %lib%   Value => ti:als:

A parameter min value of **%lib%partname** will equate to **ti:als:partname**. Now you could define %lib% to be **mot:ls:** and the example will evaluate to **mot:ls:partname**. Assuming that parameter names remain the same between libraries and the appropriate libraries are on the current library list, you can use this technique to switch manufacturers and logic families.

Chapter 6: Timing Analysis and Measurement Functions

This chapter is devoted to functions that help the user measure times, view diagrams, and improve efficiency.

Measurement Features: The Time Readout buttons and Time Markers are objects used primarily for measurement purposes. Additionally, many of the objects in the timing diagram can be configured to display or measure the times relevant to those objects.

6.1 Time Readouts, Delta Measurements and Fast Scrolling

6.2 Time Markers for Measurement

6.3 Measurement Capabilities Overview

Viewing and Hiding the Timing Diagram: The timing diagram editor supports several different zoom methods including centered zooming and drag-and-drop zoom. For viewing and documentation purposes, parts of the timing diagram can be hidden or compressed by using a special type of time marker.

6.4 Zooming Features

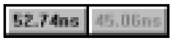
6.5 Time Markers for Time Breaks and Compression

Efficiency Features: There are many features like drag-and-drop file loading, roll-up dialogs, and a dedicated delete mode that we have added just to make the program easier to use.

6.6 User Efficiency Features

6.7 Delete Mode

6.1 Time Readouts and Delta Measurements (plus fast scrolling)



The two buttons directly above the signal label window provide both an absolute time readout and a relative time readout. The **Time** button, on the left and with the black writing, displays the current position of the mouse cursor in the Diagram window. The **Delta** button, on the right with the blue writing, displays the difference between the mouse cursor and the delta mark (an upside-down, blue triangle) on the timeline above the Diagram window.

Measurement: Every time the left mouse button is released in the Diagram window the delta mark moves to that location. To measure the distance between two objects (two signal transitions for example):

- Select an object by left clicking on it. The delta mark will move to that location.

- Move the cursor to the other object. The delta button will display the exact time between the two events.

Scrolling: The **Time** and **Delta** buttons can also be used for quick scrolling in very long timing diagrams.

- Left clicking on either button opens an edit dialog that accepts time values.
- Entering a value in the **Time** button causes the Diagram window to scroll to that exact time.
- Entering a value in the **Delta** button causes the Diagram window to scroll that amount from its current position.

6.2 Time Markers for Measurement

The basic documentation time markers are vertical lines which display their placement time in the timing diagram. This topic describes how to add, move, and edit markers. Once a marker is added to the timing diagram its behavior can be altered by changing the **Marker Type** control in the *Edit Time Marker* dialog. These advanced behaviors are covered in the following chapters:

- **Timebreak (dots), Timebreak (curved)**, and **Timebreak (jagged)** time markers are used to compress time (*Chapter 6: Time Markers for Time Breaks and Compression*).
- **End Diagram** time markers are used to indicate the effective end of the timing diagram for simulation and stimulus generation.
- **Loop Start**, **Loop End**, and **Exit Loop When** are used to specify test bench loops for TestBencher Pro (*see TestBencher Pro manual for more information*).
- **HDL Code** markers are used to execute a specific HDL code segment to be executed when the time marker is reached. (This is a TestBencher Pro feature.).

To add a time marker:

- Press the **Marker** button to turn it red.
- Decide if you would like to attach the marker to a fixed time or an edge.

  If a marker is *attached to a fixed time*, it will stay at that time. If a marker is *attached to an edge*, the marker will stay a fixed distance from that edge.

- If attaching to a time, make sure a signal edge or segment is not selected by selecting a non-edge object like a parameter or a signal name.
- If attaching to an edge, select that edge by left clicking on it.
- Move the cursor to the signal level for where the marker will display its time.
- Right click to add a time marker.

To move a time marker:

- Drag and drop the time marker line.

To edit a time marker or change the attachment:

- Double left click on the time marker line to open the *Edit Time Marker* dialog.
- Select the radio button that describes the new type of attachment that you want to make.
- If attaching to a time, you can enter an exact time into the edit box to the right of the radio button.
- If attaching to an edge, you can enter a relative offset from the edge that you select.
- Press the **OK** button to close the dialog.
- If attaching to an edge, the program will be in a continuous selection mode. Move the mouse so that the selection is on the desired signal edge.
- Left click down on the edge to attach the marker.

Snap signal ends to Marker (Documentation Feature): Double click on a marker to open the *Edit Time Marker* dialog and experiment with this feature:

- Check the **Signal ends snap to marker** check box and click **OK** to close the dialog. Now drag-and-drop the marker and notice that the ends of all the drawn signals snap to the new position of the marker.

6.3 Measurement Capabilities Overview

There are many different methods for measuring and displaying the time at which transitions occur. All these methods are discussed under sections relating to the types of objects they involve, but they are grouped here for easy reference and comparison.

Display signal states at a given time: Left click in the timeline in the Diagram window to drop a temporary marker line that displays the numerical state of each signal.

View the exact placement of an edge: Double clicking on a edge to open the *Edge Properties* dialog and display the min and max time of the transition (*Chapter 5: Edge Properties*).

Display the position or uncertainty of an edge: Left click on the **Text** button. Left click on an edge, right click to open an edit box. Enter the **%m**, **%M**, or **%u** control codes to display min transition, max transition, or uncertainty region (*Chapter 7: Using Control Codes in Text Objects*).

Display the distance between two transitions: Add a delay, setup, or hold parameter between the two transitions. Double click on the parameter to open the *Parameter Properties* dialog, and choose the **Distance** radio button (*Chapter 4: Parameter Properties*).

Measure the distance between two objects: Left click on the first object. This moves the blue delta mark, the blue triangle on the timeline, to that position. Now the blue Delta button above the signal names will display the distance between the blue delta mark and the mouse cursor. Put the mouse cursor over the other object and the distance will be displayed in the blue readout above the signal label window (*Chapter 6: Delta Measurements*).

Display the exact placement of a time marker: Left click on the **Marker** button. Right click in the diagram to add a time measure line. Double left click on the marker line to open the *Edit Time Marker* dialog and edit the exact position of the marker (*Chapter 6: Markers*).

Visual display of critical paths: Delays are color coded to indicate which edges of the delayed transition they control. This is particularly useful in diagrams where multiple delays force a single signal transition (*Chapter 4: Delays* and *Chapter 5: Multiple Delays*).

6.4 Zoom Features

The Zoom buttons, located on the right of the button bar, change the zoom level in the timing diagram. The Zoom functions are also available in the View menu. There are three main ways to change the zoom level of the diagram:

1) Use the **Zoom In** and **Zoom Out** buttons on the button bar to center and zoom on a selected object. For example:

- If an object is selected in the Diagram window, then the zoom will center on that object.

- If there are no selected objects, but the Blue Delta Triangle is visible, then zoom will center on the Blue Delta Triangle. To move the Blue Delta Triangle, click at the desired time location in either the time line or the diagram window.

- If no edge is selected and the blue triangle is not on the screen, then the diagram zooms from the center of the Diagram window.

2) **Click-and-drag on the Time Line.** This provides a quick way to graphically specify the zoom level and range for a section in a large timing diagram.

- *Click* inside the timeline and *drag* the mouse to indicate the range in which to zoom.

- When you release the mouse the diagram will zoom to show the range you selected.

3) Use the **Zoom Range** or **Zoom Full** buttons on the button bar.

- The **Zoom Range** button opens a dialog that lets you specify the starting and ending times displayed in the Diagram window.

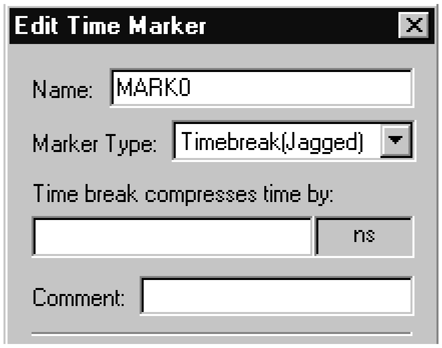
- The **Zoom Full** button displays the entire timing diagram.

6.5 Time Markers for Time Breaks and Compression

Time Breaks are a special type of time marker that can compress time. The time breaks can be dotted, curved, or jagged double lines that mimic the most common time breaks used in data books. If a compression time is added to the time break then that amount of time to the right of the marker will not be displayed on the screen. The compressed time still exists, but it does not display on the screen.

To add a time break:

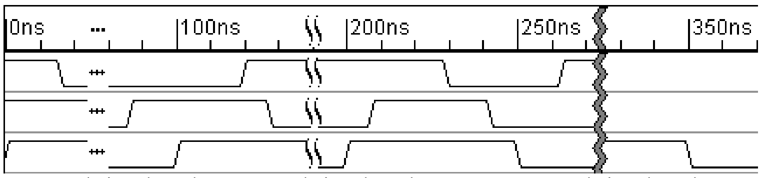
- Add a regular time marker. Press the **Marker** button and use the right mouse button to add a time marker (*Chapter 6: Time Markers*).
- Double click on the marker line (not on the marker name) to open the *Edit Time Marker* dialog.



Name: MARK0

Marker Type: Timebreak(Jagged)

- From the **Marker Type** drop down list box, choose one of the three graphical time break styles: **Timebreak(dots)**, **Timebreak(curved)**, or **Timebreak(jagged)**.
- Use the **Time break compresses time by:** edit box to indicate the amount of time that should be compressed. The time to the right of the marker will be compressed and will not display on the screen. The compressed time still exists, but it does not display on the screen. If no time compression is specified, then the time break will act as a graphical display, and no time will be compressed. (Note: this edit box appears when you select one of the three **Timebreak** marker types.)
- Click the **OK** button to close the dialog.



Dotted timebreak Curved timebreak Jagged timebreak

6.6 User Efficiency Features

Here is a short list of features we added just to make the program easier to use:

- **Multiple Undo and Redo:** The **Edit > Undo** *action* and **Edit > Redo** *action* menu options describe the type of editing or deleting action that you can undo. The older **Edit > Undo Delete** menu option remains as a menu option so that deletions can be undone without changing the multiple undo/redo buffer.
- **Roll-up Dialogs:** The modeless dialogs like *Signal Properties* and *Parameter Properties* can be rolled up in order to view the timing diagram. To roll up a dialog, press the minimize button in the top right hand side of the dialog window bar. When a dialog is rolled up only

the window bar is visible. To unroll a dialog press the maximize window button on the window bar.

- **Cursor Shape Indicates Editing Operations:** The cursor changes shape to indicate that different editing options are available when the cursor is over a particular object.

Object	*Cursor*	*Action*
parameter	up-down arrow	can move parameter up or down
parameter handle	pointer	drag handle to a new edge
text	four-way arrow	drag or nudge in any direction
marker or edge	left- right arrow	drag or nudge left or right
all other objects	type of waveform to be drawn next	mirrors the red state button

- **Proportional Scroll Bars:** The thumbs on the scroll bar are now proportionally sized to indicate how much of the file is being viewed.
-**Compact Dialogs:** Small, compact dialogs allow maximum use of your screen real estate (great for laptops).
- **Drag-and-Drop File Load:** From Windows Explorer you can drag a *.tim file (the program's normal output file) and drop it into the timing window to open the file. This is helpful if you have many timing diagrams and would like to view them in rapid succession.
- **File History List:** The File menu has a history list of the last four files that were used. This helps you quickly load your current project. Left click on a file in the history list to open that diagram.
- **Intelligent Windows:** The Diagram and Parameter windows remember their size and location the last time the program was run. This means that you don't have to size and move the windows each time you run the program. Fonts, colors, and window specific options are also remembered.
- **Modeless Drawing:** SynaptiCAD products allow you to draw and edit waveforms regardless of the mode you happen to be in.
- **Status bar and Quick Overviews**: We have status bars and a short overview to help new users get started.
- **Edit Box History Lists:** Many edit boxes are equipped with permanent history lists that store information from session to session. History lists save the user from retyping previously entered information. History lists are stored in an **.ini** file in your Windows directory (**timing.ini** for Timing Diagrammer Pro, **waveform.ini** for WaveFormer Pro, and **tbench.ini** for TestBencher Pro) and can be pruned and rearranged in that file (be sure the program is not running while editing its **.ini** file or your changes will be overwritten by the program when it is closed).

6.7 Delete Mode

The Delete Mode lets you quickly delete objects in the Diagram window without having to use the keyboard.

To enter Delete Mode:

- Check the **Edit > Right Click Delete Mode** menu option.

  Note: When the Delete Mode is active, the other mode buttons are gray and inactive.

- Select an object by left clicking on it.
- Right click to delete the object. Remember that the right mouse performs the function of the activated mode button.

If you make a mistake, the **Edit > Undo** and **Edit > Undo Delete** menu options can return the diagram to its original state.

Alternate Method: Objects can also be deleted by selecting them and then pressing the **<Delete>** key on the keyboard or selecting the **Edit > Delete** menu option.

6.8 Specifying a User-defined Radix

The radix used to display state values on a signal can now be specified as a user-defined radix type. This allows you to specify easily recognizable text to be displayed in place of the binary value held by a state. For example, you can determine that the words "True" and "False" will be displayed in place of 0 and 1.

To Specify a User-Defined Radix:

- Select the **Bus > Create User-Defined Radix** menu option. This will open the *Create User-Defined Radix* dialog box.
- Enter the name you want the radix to have (e.g., Boolean) in the **Name** edit box.
- Click the **Add** button below the **Name** edit box to add the new radix type.
- Enter the numerical value in the **State Value** edit box or select the value from the drop down list-box.
- Enter the corresponding text in the **Translation** edit box.
- Click the **Add** button in the **State Translation** area of the dialog box. This will add the new state translation to the new radix type.
- Repeat the last three steps until all desired values have been added to the radix.
- Click the **OK** button to close the *Create User-Defined Radix* dialog box.

A User-Defined Radix can be specified as the default radix for new signals by selecting the radix name from the **Default Radix** drop-down list box in the *Create User-Defined Radix* dialog box.

To add new state translations to a User-Defined Radix:

- Select the **Bus > Create User-Defined Radix** menu option. This will open the *Create User-Defined Radix* dialog box.
- Select the name of the radix you want to modify (e.g., Boolean) in the **Name** drop-down list box.
- Enter the numerical value in the **State Value** edit box or select the value from the drop down list-box.
- Enter the corresponding text in the **Translation** edit box.
- Click the **Add** button in the **State Translation** area of the dialog box. This will add the new state translation to the new radix type.
- Repeat the last three steps until all desired values have been added to the radix.
- Click the **OK** button to close the *Create User-Defined Radix* dialog box.

Chapter 7: Text Objects

Text objects can be placed anywhere in a timing diagram. They can be used to annotate cycles, signal edges, or segments.

The font and color of each text object can be changed to stress the importance of that particular text object. The fonts also support superscripts, subscripts, and bold and italic text attributes so your timing diagrams can use the same names and comments that are commonly used in data books.

Text objects automatically snap to an invisible alignment grid that makes it easy to visually line up different objects.

7.1 Adding Text

There are three ways of setting the horizontal position of a text object. First, text can be **attached to a specific time** so that the text stays at that same time regardless of how the timing diagram changes. Second, text can be **attached to an edge** so that the text moves when the edge is moved, keeping the same fixed distance from that edge. And third, text can be **attached to a segment** so that the text stays centered within that segment.

To add a text object:

- Press the **Text** button to turn it red.
- Decide if you would like to attach the text to a fixed time, an edge, or a segment and select the appropriate object.

  a) **Time (most common):** Any object except a signal or edge.
  b) **Edge:** Select the edge by left clicking on it.
  c) **Segment:** Select the segment by left clicking on it.
- Right click to open an edit box.
- Type in the text. **Note:** This is a rich edit control that supports bold, italic, subscript, and superscripts. To apply these styles, select the text and use the following key combinations. Or before typing the text, use the key combinations to turn on the style so the text you type is formatted that way. The key combinations are:

  Bold the text using **<Ctrl>-b** keys
  Italicize with **<Ctrl>-i** keys
  Superscript with **<Ctrl>-u** keys (u for up)
  Subscript with **<Ctrl>-d** keys (d for down)
- Close the edit box by pressing the **<Enter>** key or clicking outside of the box.

Multi-line text objects are constructed by editing a single line text object using the *Edit Text* dialog.

7.2 Editing Text

A text object's properties, text, attachment, fonts, and color are edited in the *Edit Text* dialog.

To open the *Edit Text* dialog:

- Double left click on the text object.

OR

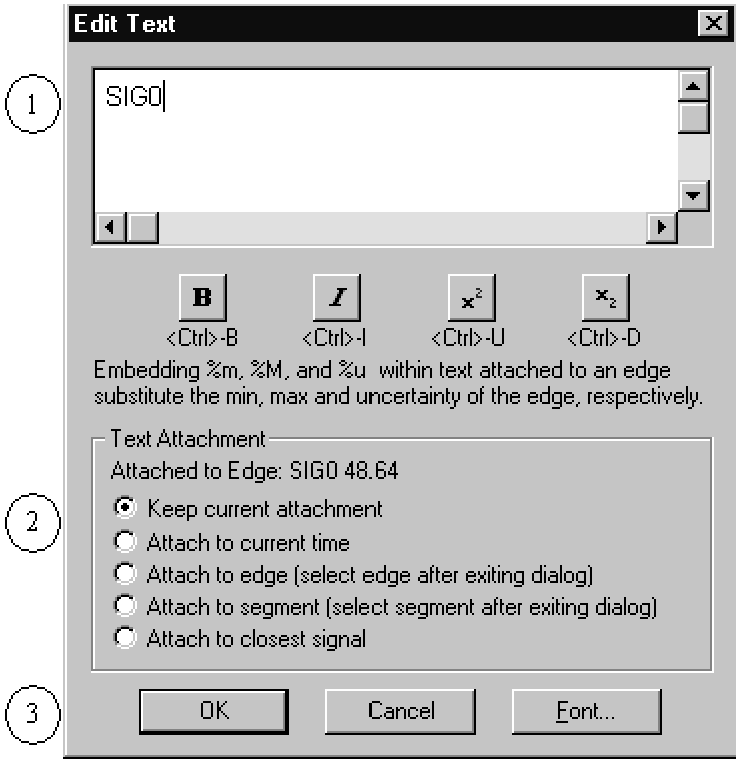
- Select the text (left click), and choose the **Edit > Edit Text** menu item. This menu is only enabled when a text object is selected.

There are three sections in the *Edit Text* dialog (See Figure 7.1).

1) Text Edit Section: The edit box at the top of the *Edit Text* dialog accepts text from the clipboard, multiple lines of text, and special style editing:

- **Multi-line text:** Paragraph text blocks can be added by typing in multiple lines of text into the edit box. By default, new text objects are single-line so that you can add many in rapid succession.

- **Copy and Paste text:** The normal Windows copy and paste keystrokes (**<Ctrl>-C** and **<Ctrl>-V**) can be used inside this edit box to transfer text to and from word processors and other editors.



SIG0

Embedding %m, %M, and %u within text attached to an edge substitute the min, max and uncertainty of the edge, respectively.

Text Attachment

Attached to Edge: SIG0 48.64

- [✓ Keep current attachment]
- Attach to current time
- Attach to edge (select edge after exiting dialog)
- Attach to segment (select segment after exiting dialog)
- Attach to closest signal

Figure 7.1 Edit Text Dialog Box

- **Bold, Italic, Superscript, and Subscript Styles:** To apply these styles, select part of the text and press one of the style buttons located below the edit box.

2) Text Attachment Section: Text objects are either attached to a fixed time, an edge, a segment, or a signal. To change the attachment of a text object:

- Select the radio button that describes the new type of attachment that you want to make.
- Choose the **OK** button to close the *Edit Text* dialog.
- If you chose **Attach to current time,** then the text will be fixed at that time.
- If you chose **Attach to edge**, **Attach to segment**, or **Attach to signal**, the timing diagram editor will be in a continuous selection mode.
- Move the mouse so that the selection is on the desired signal edge or segment.
- Left click on the edge or segment to attach the text.

3) Font and color section: When a text object is first drawn it uses the default font for the Diagram window (see **Options > Text/Color Preferences > Set Diagram Window Font** menu option). Each text object can have a totally different font, size, and color. To change the font or color:

- Press the **Font** button to open the *Font* dialog.
- Choose the desired **font**, **size**, and **color** from the dialog. **Note:** The font style, strike out, and underline controls of the of the *Font* dialog are <u>not</u> supported and do not affect the text object.

7.3 Moving Text and the Alignment Grid

Text can be moved using the mouse or the keyboard. Text which is attached to a time or an edge can be moved in all four directions (left, right, up, down). Text which is attached to a segment can only be moved up or down, because the horizontal position is determined by the segment.

Drag-and-Drop text with the mouse:

- Left click on the text object and drag it to the desired location.
- Release the mouse button to drop the text.

Nudge the text using the keyboard:

- Select the text by left clicking on it.
- Use the arrow keys on the keyboard to nudge the text up, down, left, or right.

The placement of text objects is controlled by an invisible text alignment grid which makes it easy to align text and make organized timing diagrams. As you drag or draw a new text object, it is pulled into alignment with the nearest intersection of grid lines.

For text objects, the default grid settings are .001 display time units in the horizontal direction and 6 pixels in the vertical direction. This setting makes it easy to vertically line up text, but leaves complete placement freedom in the horizontal direction.

Change the Text alignment grid:

- Select the **Options > Grid Settings** menu option to open the *Edit Text and Edge Grids* dialog.
- Enter the new horizontal and vertical values.
- Click the **OK** button to close the dialog. All new text objects will be aligned on the new grid intersections. Existing text objects will hold their current positions.

7.4 Using Control Codes in Text Objects

The special control codes **%m**, **%M**, and **%u** allow text objects to display the min, max, and uncertainty times of the transition that is attached to the text object. The text object can contain any string of characters including these control codes. When the program detects these control codes it replaces them with the transition's time values. All other text is displayed as a standard text object.

To use the control codes in a text object:

- Select an edge in the diagram (edges with uncertainty are more interesting to experiment with).
- Add a text object and type in the following text string

  min = %m, max = %M, and uncertainty = %u

- Press the **<Enter>** key to close the edit box. Depending on the position of the edge, the text string should look something like: **min = 50, max = 60, and uncertainty = 10**
- Notice that the ordinary text is displayed along with the time values of the transition.
- If you can still see the control codes then the text object is not attached to an edge.

7.5 Deleting Text Objects

To delete a text object:

- Select the text object (left click on it).
- Type the **<Delete>** key on the keyboard, use the **Edit > Delete** menu option, or use the right mouse button while in **Delete Mode**.

Chapter 8: Timing Diagram Formatting

Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro and TestBencher Pro have many different features for changing the way a timing diagram looks on the screen and how it prints to a file. In particular, waveforms, parameters, and signal names have a variety of features that affect how those objects are displayed. Each of these is covered in its own section. The remainder of the chapter describes how to change the fonts, colors, diagram window, and parameter table displays.

8.1 Waveform Display

Waveforms can be made to snap to an edge grid or end at the same point in time. The color, line thickness, and edge slant can be controlled using the *Drawing Preferences* dialog.

1) To make all waveforms end at a common time:

- Add a time marker to the diagram (*Chapter 6: Time Markers for Measurement*).
- Double left click on the time marker line to open the *Edit Time Marker* dialog.
- Check the **Signal ends snap to marker** check box and click **OK** to close the dialog.
- Note that the waveforms snap to the marker, but are not truncated. All signal transitions remain intact, even if the marker is moved. Only the last segment of the signal ends at the marker.

2) The **Edge Alignment Grid** helps you place signal edges at exact times. As you drag or draw an edge, it is pulled into alignment with the nearest intersection of the alignment grid. The grid does not affect the placement of edges that are moved by delays or formulas. The default grid setting is one display time unit in the horizontal direction. This setting makes very nice VHDL and Verilog stimulus generation files because all the transitions line up on whole number times (2ns instead of 2.465ns for example). Sometimes it is convenient to set the grid to a multiple of the clock frequency to make all new signal edges line up with the clock edges. To change the Edge Alignment Grid:

- Choose the **Options > Grid Settings** menu item to open the *Edit Text & Edge Grids* dialog.
- Enter the new horizontal value in the **Signal Edge Grid** section.

- Click the **OK** button to close the dialog.

- All new edges and edges that are moved will automatically be aligned on the new grid.

3) General Waveform Display: Color, Thickness, Edge Slant, and Line Separators affect how the waveforms are drawn on the screen.

- Choose the **Options > Drawing Preferences (Style Sheet)** menu to open a dialog of that name.

- Make appropriate changes and click **OK** to close the dialog.

8.2 Parameter Display

Delay, setup, hold, and sample parameters can have data-book names that include superscript and subscript text. The vertical placement can be changed by dragging and dropping the parameters. The display of each parameter in the diagram window can be customized to display any string or combination of parameter values.

1) **Realistic Data-Book Parameter Names** allow users to produce publication quality timing diagrams for inclusion in data books and application notes. The following styles are supported in all text objects, comments, min/max formulas, and parameter names: subscript, superscript, bold and italics. For example, valid parameter names include:

t_{pd}, **OffDelay**, *OnDelay*, t^{up}

Formatting styles are ignored during formula evaluation. The names tpd, **tpd,** and *tpd* are treated as the same parameter. To add formatting styles to a parameter name:

- Double click on the parameter to open the *Parameter Properties* dialog.

- Select all or part of the name to format.

- Use the following keys to toggle the text formatting: bold = **<Ctrl>-b**, italicize = <**Ctrl>-i**, superscript **<Ctrl>-u** (u for up), subscript = **<Ctrl>-d** (d for down).

2) **Parameter Vertical Placement:** Normally the timing diagram editor determines the best vertical placement for parameters on the screen, but the user can also place parameters at a specific height or between specific signals. To change the placement of a parameter:

- Left click on the center of a parameter and drag it to a new vertical position.

Note: Depending on the amount of dragging, the parameter moves either one vertical position in the original signal space or to a space between two new signals.

After you move a parameter it is considered **user placed** and it will not be moved from that position unless moved by the user. To return placement control to the timing diagram editor:

- Double left click on the parameter to open the *Parameter Properties* dialog.

- Uncheck the **user placed** check box and click **Apply** or **OK** to apply the change.

3) **Parameter Display Label Defaults and Customization:** The global and individual *Display Label* settings control how a parameter displays itself in the diagram window. When a parameter is added to the diagram window, it will display the value determined by the *Global Display Label* setting. However each parameter can override the global setting by using its own individual *Display Label* setting. To change the individual or global setting:

- **For an individual parameter:** Double click on the parameter to open the *Parameter Properties* dialog.

OR

- **For the Global parameter setting:** Choose the **Options > Drawing Preferences** menu option to open the *Drawing Preferences* dialog.

- Use the **Default Parameter Display** drop-down list box to choose the display value. The following attributes may be displayed:

- **Name, min/max value, min/max formula, min/max Margin, and comment:** These will display the attribute as it appears in the parameter window.

- **Distance:** Displays the minimum and maximum distance between transitions. This value takes into account reconvergent fanout effects.

- **Custom:** Displays a custom string of characters and attributes. Certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a **%** character followed by one or two letters. The control codes are:

- name = **%n**		
- min value = **%mv**	and	max value = **%Mv**
- min formula = **%mf**	and	max formula = **%Mf**
- min margin = **%mm**	and	max margin = **%Mm**
- min distance = **%md**	and	max distance = **%Md**
- comment = **%c**		

- Check the **Outward Arrows** checkbox to display the setup or hold with the traditional data book format of outward pointing arrows. Unchecked displays the default "point to control signal" style arrows. Outward arrows are useful when using setups as distance indicators or pulse width checks.

- Check the **Auto place Setups/Holds on lower signals** to make setups and holds appear above the lowest attached signal. If not checked then they appear above the highest attached signal.

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

8.3 Signal Name Display

The *Label window* displays the signal names. By default the Label Window is wide enough to display about 22 characters and is left justified. Both the width of the Label Window and the text justification can be altered. Also, the font and color of the window can be changed. Signal names displayed with a bar over the name indicate an active low signal.

1) To change the width of the *Label window*:

- Drag and drop the bar separating the signal names and signal waveforms.

2) To change the Auto Name Generation:

- Right mouse click on the **Add Signal** button on the button bar. This will open the *Modify Auto Name Prefix* dialog.
- Type in the new prefix and press **OK**.

3) To right justify the signal names:

- Select the **Options > Drawing Preferences** menu option to open the *Drawing Preferences* dialog.
- Check the **Right Justify Signal Names** checkbox.
- Click the **OK** button to close the dialog.

4) To make signal names active low:

- Double click on the signal name to open the *Signal Properties* dialog.
- Click the **Advanced Register** button to open the *Advanced Register and Latch Controls* dialog.
- Either check the **Active Low** check box, or type **$BAR** on the end of the name.

5) To change the font used for the signal names:

- Choose the **Options > Text & Color Preferences > Set Signal Label Font** menu option to open the *Font* dialog**.**

6) To change the background color of the Label window:

- Choose the **Options > Text & Color Preferences > Set Signal Label Background Color** menu option to open the *Color* dialog**.**

8.4 Marker Display

Markers can be made to display different values by using the *Edit Time Marker* dialog or the *Drawing Preferences (style sheet)* dialog.

Marker Label Defaults and Customization: The global and individual *Display Label* settings control how a marker displays itself in the diagram window. When a marker is added to the diagram window, it will display the value determined by the *Global Marker Display Label* setting. However each marker can override the global setting by using its own individual *Display Label* setting. To change the individual or global setting:

- **For an individual marker:** Double click on the marker to open the *Edit Time Marker* dialog.

OR

- **For the Global marker setting:** Choose the **Options > Drawing Preferences (style sheet)** menu option to open the *Drawing Preferences* dialog.

- Use the **Default Marker Display** drop-down list box to choose the display value. The following attributes may be displayed:

Default Marker Display:

Display Label: This setting controls the default label used for displaying markers in the drawing window. The following attributes may be displayed:

- **Name, min value, min/max value:** will display the attribute as it appears in the *Marker* dialog.
- **Distance:** displays the total time compressed.
- **Type:** displays the type of marker (eg. documentation, loop start, loop end, …). This is useful in debugging Marker-generated HDL code.
- **Custom:** displays a custom string of characters and attributes. Certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a % character followed by one or two letters. The control codes are:
- name = **%n**
- min value = **%mv** and max value = **%Mv**
- min distance = **%md** and max distance = **%Md**
- type = **%t**
- comment = **%c**

Note: Individual markers can be made to display a different attribute by using the *Edit Time Marker* dialog.

8.5 Diagram Window Display

The Diagram window can be customized through a variety of methods. Spacers can be added to insert a space into timing diagrams for aesthetic purposes. Individual and groups of objects can be hidden. Fonts, background color, and display options can be set using controls under the Options menu.

1) To insert a horizontal spaces into the diagram:

- Left mouse click on the **Add Spacer** button.

2) To convert a regular signal into a spacer:

- Double left click on the signal name of a signal with NO waveforms drawn. This will open up an edit box with the signal name selected. If the signal contains waveforms they must first be deleted before turning it into a spacer.
- Backspace once to clear the signal name.
- Press the <Enter> key or mouse click outside of the edit box to close the edit box. The signal is now a spacer.

3) To hide individual signals or parameters:

- Select the signal or parameter by left clicking on the name of the object.
- Select the **View > Hide Selected Signals** or **View > Hide Selected Parameter** menu option to hide the selected object.

4) To hide groups of signals and parameters using filter patterns:

- Select the **View > Filter Signals** or **View > Filter Parameters** menu option to open the filter dialog.
- Enter a pattern and press the **Add** button to activate it. See section 1.7 for more information.

5) To hide all objects of a given type:

The **View** menu has several **Show *object*** sub-menus that can be used to hide entire sets of objects like all delays or all grid lines. In particular, two of the show options are important to documentation users:

- **Show Hidden Text** allows attachments to signals like text objects and grid lines to continue to be displayed even when their parent signal is hidden. This lets you create data-book quality diagrams, by using signal segments to line-up and center text without having to show the signal segment.
- **Show Critical Paths** is the option that lets delays change color to indicate which edges of the delayed transition are set by the particular delay. This is an important feature for design. But for documentation, you can uncheck the menu option to make all the delays black.

6) The **General Diagram Window Options** affect how numbers and parameters are displayed, and how waveforms are drawn on the screen.

- Choose the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog and set the truncate trailing zeros and hide small parameter labels controls.
- The Color and Fonts can be changed using the **Options > Text/Color Preferences > Set Diagram window** menu options

8.6 Parameter Table Display

The Parameter Table displays parameters in a convenient databook format. Every time a new parameter is added to the Diagram window an entry is made in the parameter table. To edit the values in the parameter table, double click on the parameter and use the *Parameter Properties* dialog box. The data inside the Parameter Table can be rearranged and viewed with the following functions:

To adjust the width of a column:

- Move the cursor to the right-hand side of the column heading. When the cursor changes to a double arrow, left click on the line and drag it to the new width of the column. Place it by releasing the left mouse button.

To move a parameter:

- Left click on a cell in the parameter to be moved.
- Put the mouse cursor near the very top or bottom of the parameter. When the mouse cursor changes from a normal arrow to an up/down double arrow, press the left mouse button down. A green bar will appear.
- Drag the green bar to the new location and drop it by releasing the left mouse button.

To hide a parameter in the Parameter window:

- Left mouse click on the parameter's name.
- Select the **View > Hide Selected Parameter Data** menu option.

Hiding a parameter in the parameter table will hide its timing data and all instances of the parameter in the Diagram window. The parameter will continue to function normally, however timing errors will not be shown until you unhide the parameters. To show a hidden parameter choose the **View > Show Hidden Parameter Data** menu option. This opens a dialog box that allows you to selectively unhide parameters.

To delete a parameter from the Parameter Table:

- Select the parameter by left clicking on any cell of the parameter.
- Then select the **Edit > Delete** menu option OR press the <**delete**> key.

To switch between min/max value and min/max formula display:

- Select either the **Options > Parameter Window Preferences > Display min/max value** menu option or the **Options > Parameter Window Preferences > Display min/max formula** menu option. These are mutually exclusive parameter window options; the default is by value.

8.7 Drawing Preferences Dialog

The *Drawing Preferences* dialog controls how objects appear in the Diagram window. To open this dialog choose the **Options > Drawing Preferences** menu option in the timing window. Below is a list of the options that can be set with this dialog (see Figure 8.1):

Signal Display Settings

Signal Label Font: Opens font dialog. You can change the font, size, and color of the signal labels.

Signal Color: Signals waveforms can be drawn in any color. Default is black.

Track Font: Keeps waveform height relative to the signal label height.

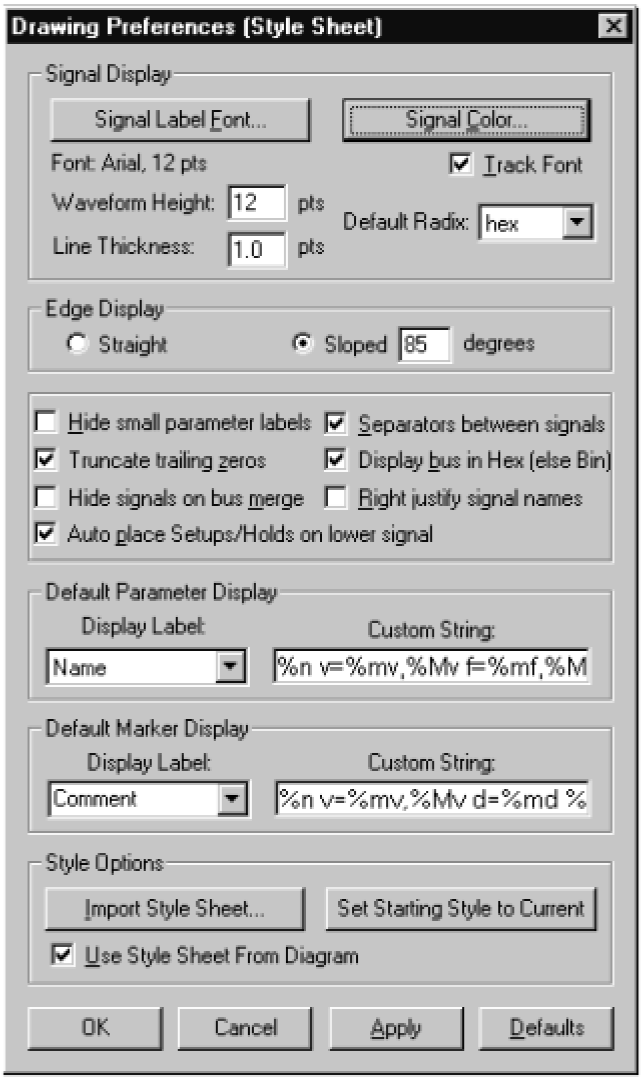
Waveform Height: Sets the height of the waveform.

Line Thickness: Signal waveforms can be drawn with different line widths. Default is 1 point (1/72th of an inch).

Default Radix: Default radix used to display virtual states on new signals.

Edge Display Settings

Sloped or **Straight:** Signal transition edges can be drawn as sloped or straight lines depending on which radio button is selected. Default is sloped edges. The angle of the slope can also be controlled (the default is 2 pixels).



Signal Display

Font: Arial, 12 pts

[✓] Track Font

Waveform Height: 12 pts

Line Thickness: 1.0 pts

Default Radix: hex

Edge Display

Straight [✓ Sloped] 85 degrees

[ ] Hide small parameter labels

[✓] Separators between signals

[✓] Truncate trailing zeros

[✓] Display bus in Hex (else Bin)

[ ] Hide signals on bus merge

[ ] Right justify signal names

[✓] Auto place Setups/Holds on lower signal

Default Parameter Display

Display Label: Name

Custom String: %n v=%mv,%Mv f=%mf,%M

Default Marker Display

Display Label: Comment

Custom String: %n v=%mv,%Mv d=%md %

Style Options

[✓] Use Style Sheet From Diagram

Figure 8.1 Drawing Preferences Dialog Box

General Display Settings

Truncate trailing zeros: Removes trailing zeros after the decimal point in time values (makes the display more readable). Default is checked.

Hide small parameter labels: If checked, the parameter labels will be hidden if the parameter is visually small (less then 2 pixels wide). When zooming out to a larger time scale, parameters shrink, but labels stay the same size. This option will prevent these small parameter labels from cluttering the screen. Default is unchecked.

Hide signals on bus merge: Bus member signals can be automatically hidden when a bus is created.

Auto place Setups/Holds on lower signals: If checked setups and holds appear above the lowest attached signal. If not checked then they appear above the highest attached signal.

Separators between signals: Signals can be displayed with and without separator lines depending on the state of the checkbox. Default is "with separator" lines.

Display bus in Hex (else Bin): buses can display values either in Hex or Binary.

Right justify signal names: If checked, the signal names in the Label window will be right justified.

Default Parameter Display:

- Display Label: This setting controls the default label used for displaying parameters in the Diagram window. The following attributes may be displayed:

- **- Name, min/max value, min/max formula, min/max Margin, and comment:** will display the attribute as it appears in the parameter window.
- **- Distance:** displays the minimum and maximum distances between transitions. This value takes into account **reconvergent fanout** effects covered in Chapter 5.
- **- Custom:** displays a custom string of characters and attributes. Certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a % character followed by one or two letters. The control codes are:

- name = **%n**		
- min value = **%mv**	and	max value = **%Mv**
- min formula = **%mf**	and	max formula = **%Mf**
- min margin = **%mm**	and	max margin = **%Mm**
- min distance = **%md**	and	max distance = **%Md**
- comment = **%c**		

Note: Individual parameters can display a different attribute by using the *Parameter Properties* dialog.

Default Marker Display:

Display Label: This setting controls the default label used for displaying markers in the drawing window. The following attributes may be displayed:

- **Name, min value, min/max value:** will display the attribute as it appears in the *Marker* dialog.
- **Distance:** displays the total time compressed.
- **Type:** displays the type of marker (eg. documentation, loop start, loop end, …). This is useful in debugging Marker-generated HDL code.
- **Custom:** displays a custom string of characters and attributes. Certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a % character followed by one or two letters. The control codes are:

- name = **%n**
- min value = **%mv** and max value = **%Mv**
- min distance = **%md** and max distance = **%Md**
- type = **%t**
- comment = **%c**

Note: Individual markers can be made to display a different attribute by using the *Edit Time Marker* dialog.

Style Sheet Settings:

Use Style Sheet From Diagram: If checked, when you open an existing timing diagram the program will use the style information stored in the current timing diagram. If it is unchecked then the current style information will override the style information stored in the timing diagram. Default is checked.

Import Style Sheet: Opens a file dialog that lets you load the style information from another timing diagram.

Starting Style to Current: Clicking this button will make the current style settings become the default settings for new timing diagrams.

8.8 Font and Color Settings

Text/Color Preferences is a sub menu of the Options menu. All project wide fonts and colors are set using this menu. If you want to change the font of a specific text object, see *Chapter 7: Text Objects*. The following changes can be made using this sub-menu:

Set Global Default font

Set Diagram Window font

Set Diagram Window background color

Set Signal Label font

Set Signal Label background color

Set Parameter Window font

Set Parameter Window background color

The **Set font** options set the font, style, and color for the three main text windows. The **Set background color** options sets the background color for these same sections. By default the Windows™ system font and Windows™ background color are used. To change the default font for all sections use the **Option > Text/Color Preferences > Set Global Default** menu option.

Set Parameter Color Strip **color**

Set Parameter Color Strip **font color**

The Parameter Color Strip is the bar in the parameter window that contains the column headings. Both the background color and the font color are user-definable.

Set Button Bar Color Strip **color**

The Button Bar Color Strip is the background color of the button bar in the main timing window.

Restore Default Style

This Option restores the default Windows fonts and background colors to the drawing, label and parameter windows. Text objects with individually specified fonts are not affected by this menu option.

Chapter 9: Images, OLE, and Printing

WaveFormer Pro, Timing Diagrammer Pro, VeriLogger Pro and TestBencher Pro support a wide variety of image generation, printer options, OLE embedding, and TDML support. There are three main groups of documentation features:

1) Image Generation: The timing diagram editor supports six different ways to convert your timing diagrams into image files which can be embedded into word processors, graphics packages, and page layout programs.

2) Image Management: If you are managing a large number of timing diagram images, then you will want use the OLE or TDML features. OLE provides a quick way to manage image files and the corresponding timing diagram files inside word processing documents. TDML provides an industry standard format for on-line timing diagrams.

3) Printer Options: Outputting timing diagrams directly to a printer is the fastest and easiest way to document large timing diagrams. These print-outs are designed to be used by engineers to document their designs.

9.1 Embedding Images into Documents

There are six different ways to convert your timing diagrams into image files. Image files can then be embedded into word processors, graphics packages, and page layout programs.

Vector files can be sized and scaled without affecting the crispness of the lines that make up the timing diagrams. Most word processors support editing of the metafile types. Not all programs support all six file types so we encourage you to experiment and find out what works best for your tool suite.

The following General Purpose image formats are supported:

Name	type	ease of use	comments
Copy-to-Clipboard	Windows bitmap	very easy	medium quality images which cannot be sized or scaled without losing resolution
WMF metafiles	vector metafile	very easy	excellent quality images which can be sized or scaled recommended for Microsoft Office/ Windows Applications
MIF files	vector file	moderately easy	For use with FrameMaker only. Read **image.doc** for more information

The following Specialized image file formats are also supported:

Name	type	ease of use	comments
TIFF files	raster	very easy	loss-less data compression, high quality
PNG files (DataSheet Pro)	raster	very easy	web-ready, loss-less data compression, high quality
JPEG files (DataSheet Pro)	raster	very easy	web-ready, lossy data compression
CGM metafiles	vector metafile	moderately easy	excellent quality images, support for cross-platform images
EPS files (Encapsulated PostScript)	vector file	most difficult	Excellent quality images, cross-platform images and data books. Read **image.doc** for advanced image instructions which are especially helpful for EPS file generation.

9.2 Copy to Clipboard

Copy-to-clipboard is the quickest and easiest way to insert an image into a word processing document. This method uses a screen capture bitmap of the current selected window (either the timing diagram or the parameter table) and puts it on the Windows clipboard. There is no equivalent method for doing this on the UNIX platforms, so UNIX users should use EPS files.

The copy-to-clipboard method also produces the lowest quality images. The resolution of the image is based on your monitor's resolution settings. The bitmap images are not scalable or editable inside word processing programs. For higher quality images use the **File > Print Diagram** menu option and choose one of the vector image formats (e.g., Windows Metafiles WMF, FrameMaker MIF, or Encapsulated Postscript EPS).

To use copy-to-clipboard:

- Arrange the Diagram window so that your timing diagram is displayed the way you want it to appear in the document.
- Select the **Edit > Copy-to-Clipboard** menu option which opens a dialog of that name.
- Choose your options then press **OK** button. This closes the dialog and places a bitmap of the timing diagram on the clipboard.

- Inside your graphics or word processing program use the **Edit > Paste** menu command to embed the image into your document.

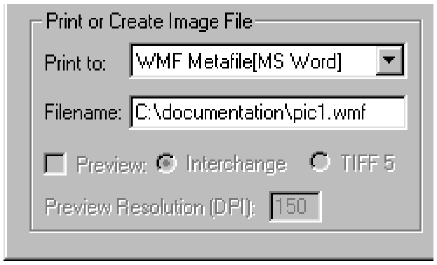
Note: For printing purposes you may want to choose the default Windows colors of black text on a white background before capturing the screen.

9.3 WMF Images for Microsoft Word

The recommended image format for Microsoft Office documents is WMF Metafiles. WMF Metafiles are the most widely supported scalable and editable image format for Windows platforms. Your timing diagrams will print with smooth lines and scaled text.

To make a WMF file:

- Select the **File > Print Diagram** menu item to open the *Print* dialog.


Print or Create Image File
Print to:
WMF Metafile[MS Word]
Filename:
C:\documentation\pic1.wmf
Preview:
Interchange
TIFF 5
Preview Resolution (DPI):
150

- Choose the **WMF Metafile [MS Word]** option from the **Print To** drop-down list box the bottom left hand corner of the *Print* dialog.

- It is not necessary to specify a filename. If you are in Windows, the *Print* dialog will automatically place the metafile on the clipboard and you can use a paste command to insert the image into a document. You may optionally specify a filename with a extension of **.wmf**.

- Choose other options then press **OK** button. This closes the dialog and places the metafile of the timing diagram on the clipboard (and creates a .wmf file if you specified a filename).

- ***Inside Microsoft Word use the* Edit > Paste *menu command*** to embed the image into a document.

Microsoft Word also supports bitmap images and EPS images, please read the file **image.doc** for more information on using these formats with Word.

9.4 MIF Images for FrameMaker

The timing diagram editor can export diagrams and parameter lists to Adobe's Maker Interchange Format (MIF). The generated MIF files can be imported into FrameMaker v3.0 and later. Inside FrameMaker, you can edit MIF images by adding extra graphic objects such as text and lines, and editing existing parts of the diagram by altering spacing, size, color, etc. The timing diagram editor supports two unique features for the MIF format:

1) The ability to set horizontal size. This is extremely useful when generating multiple MIF images of the same size (for use in a data book).

2) The ability to scale the image. Once imported into FrameMaker, the image can be scaled as desired.

To export your diagram or parameter list to MIF format:

- If printing the diagram, select **File > Print Diagram** menu option. This opens the *Print* dialog box.
- If printing the parameter table, select the **File > Print Parameters** menu option. This opens the *Print* dialog box.
- Select the **MIF File** radio button, located in the lower left hand corner of the dialog box.
- Enter a file name with a **.mif** extension in the Filename edit box.
- If you wish to set the horizontal size of the MIF file, use the check box and edit box in the **Margins section** of the dialog. Setting the image size is useful when you want to generate multiple MIF images of the same size.
- If you need to set any other options, do so now.
- Click the **OK** button to export the diagram.

To import your diagram or parameter list into FrameMaker:

- Inside FrameMaker, position your cursor at the approximate location where you would like the diagram to appear.
- Select the **File > Import > File** menu option to open the *Import File* dialog box.
- Locate your MIF file by using the dialog box. You may wish to select the **Interchange - MIF** option from the drop down combo box **List Files of Type** to help you locate the file.
- Select the **Copy Into Document** radio button. Diagrams exported from SynaptiCAD products must be imported using this option because they do not have any text flows.
- Click the **OK** button. Your diagram should now appear within FrameMaker.

9.5 Vector and Raster Image Files

Vector image formats can be sized and scaled without affecting their crispness. The following instructions are for producing images in the EPS and CGM vector formats. The file **image.doc** located on the distribution diskette has advanced image instructions which are especially helpful for EPS file generation.

- Select the **File > Print Diagram** menu item which opens the *Print* dialog.
- Choose one of the options in the **Print To** drop-down list box in the bottom left hand corner of the *Print* dialog to determine the type of file to create: CGM Metafile or EPS file.

Note: Do not select the **File** radio button. This creates a file that is only compatible with the default printer and is not a generic file type supported by other programs.

- Enter a file name with the appropriate extension: **.cgm** or **.eps**.

- Choose **OK** to create the image file.

- Refer to your word processor or graphics program documentation to find out how to embed or insert an image file into a document.

Two of the raster image types that are provided, **GIF** and **JPEG**, produce web-ready graphics. The **TIFF** and **GIF** types provide loss-less data compression, while the **Jpeg** format uses a lossy data compression. Loss-less data compression ensures, as its name implies, that no data is lost during compression. These file formats can be imported and exported using the directions above, selecting the appropriate file type.

9.6 Managing Multiple Views (Images) of a Diagram

The SynaptiCAD product line supports the concept of timing diagram **Views** which allow the storing of settings for multiple pictures of a timing diagram. Views are normally used during the documentation phase of the design when several different images will be made from one larger timing diagram. Views can also be used to save off several different zoom and scroll settings that indicate areas of interest in a large timing diagram. Views are created and edited from the *Print Diagram* and *Print Parameter* dialogs.

To create a view of a timing diagram:

- Choose either the **File > Print Diagram** or the **File > Print Parameters** menu item which opens a version the *Print* dialog.

- Change the settings in the *Print* dialog to create the desired image. The most common settings are the time range, type of image file to print, and the margin/width controls.

- Press the **Add** button to open the *Name of New View* dialog.

- Type in a descriptive name into the edit box. This is not a file name so you do not have to include a file extension.

- Press **Cancel** to exit the dialog without printing or **Print** to immediately generate an image.

To switch between views of a timing diagram:

- Open a *Print* dialog.

- Double click on the view name to load that view's printer settings.

To overwrite the settings for an existing view, press the **Add** button and use the name of the view to overwrite.

9.7 OLE - Object Linking and Embedding

Special versions of WaveFormer Pro include OLE, offering users the ability to both embed and link their timing diagrams to third-party applications. OLE provides a convenient way to manage timing diagrams which are included in many of your documents, such as word processor files, spreadsheets, and presentation graphics. Then, when you make changes to your timing diagrams, they are automatically updated in the linked or embedded third-party files. This feature greatly simplifies documentation by allowing changes to timing diagrams to be made in only one place.

Embedding can be thought of as having a separate copy of WaveFormer running inside your application software. If you have a timing diagram you want to show in a report, for instance, you may embed a copy of your WaveFormer file directly from your word processing file. You have most of the functionality of WaveFormer in the embedded object and you can edit your timing diagrams just as you normally would. Later, you can return to the embedded image by double-clicking on it, which launches WaveFormer for further editing. As you edit the timing diagram, changes are automatically shown in the embedded image in your report. You can embed as many diagrams as you want in a given document.

With linking, a single timing diagram file can be connected to many documents which need to include it. For instance, if you're working on a report and a presentation and want to show the same timing diagram in both documents, you simply link each document to the desired timing diagram file. This way, any changes you make later in the timing diagram file will be updated automatically in both your report and presentation. You may link your timing diagram to as many other documents as you like, and you can include the same link multiple times in a single document.

The special versions of WaveFormer which support OLE are technically known as OLE servers. This means that they can be linked to or embedded within other applications. These applications, in turn, are known as OLE clients since they make use of the server's features. Many of the more popular programs, like MS Word and FrameMaker, work as OLE clients. OLE is a Windows-only feature, and is not supported by UNIX X-Windows systems.

The most current information about using OLE with specific graphics programs is contained in WaveFormer's on-line help. However, the following instructions should in many cases be enough to guide you.

1) To use OLE embedding:

- Close WaveFormer since it will be launched automatically from within your third-party application.
- Run your third-party application, such as your word processor or graphics software.
- Refer to your documentation to find out how to embed an OLE object from the application you are running. We have found that there is usually a menu command resembling **Insert > Object** or **File > Import > Object** that performs this function. This menu command will open a dialog which lists all of the OLE servers available on the system.

- Choose **Timing Diagram Component** (exact wording may vary) from the list of OLE servers (usually under the **Create New** tab). This will automatically launch WaveFormer.
- Inside WaveFormer, merge in an existing timing diagram or create a new one. Notice that as you change the timing diagram in WaveFormer, the embedded image is automatically updated.
- When the timing diagram is complete you can return to your third-party application to continue your work. The embedded timing diagram remains where you left it and is saved with your document.

Your third-party application continues to display the timing diagram that you drew. If you double click on the timing diagram object, WaveFormer will launch automatically and open the diagram you just created.

2) To use OLE linking:

- Create and save a timing diagram file in WaveFormer.
- Run a third-party application, such as your word processor or graphics program.
- Refer to your third-party software documentation to find out how to link to a file. We have found that there is usually a menu command resembling **Insert > Object** or **File > Import > Object** that performs this function. This menu command will open a dialog which lists all of the OLE servers available on the system (under the **Create New** tab).
- Choose the **Create From File** tab in the dialog (instead of **Create New** which is for embedding). Then, type in the file name you want to link to. You may have to click on a checkbox which says **Link to File** to complete the dialog.
- Linking to the file automatically launches the SynaptiCAD product which created it. You are free to make changes to the file just as you would had you opened it in a stand-alone mode. However, by saving it once changes are made, you automatically update the links all other applications have with this file.
- Your application now displays the newly edited and saved timing diagram within it.

3) There is an alternative way to embed or link using cut/copy and paste:

- Run WaveFormer and create or open a timing diagram.
- Cut or copy your diagram to the clipboard.
- In your third-party application select **Edit > Paste Special**.
- In the dialog box that appears, mark the **Paste Link** radio button (for linking) or the **Paste** radio button (for embedding) and be sure the name of your application is correctly displayed in the **As** list. Click OK and you should see the linked or embedded timing diagram within your application.

9.8 TDML Management

On-line data sheets will eventually replace or augment printed data books. TDML (Timing Diagram Markup Language) is the timing diagram and timing parameter format recommended by the Si2 ECIX committee involved with the data sheet standard. With TDML you will be able to view a data sheet in a browser-like environment, then double click a timing diagram icon to automatically launch a SynaptiCAD timing diagram editor to display and manipulate the timing information.

SynaptiCAD has been working with Si2 on this format since November of 1996 and we expect to see the first on-line data sheet sites in the middle of 2000. Keep checking our web site for updates on the newest TDML information.

To create a TDML timing diagram:

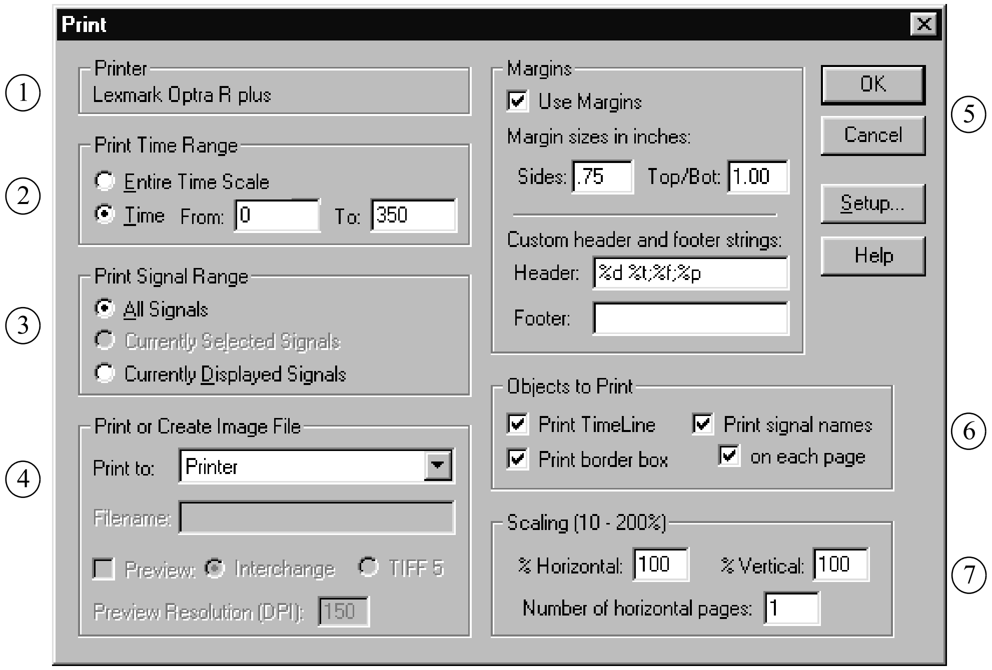
- Select the **File > Save Timing Diagram As** menu option to open the *Save* dialog.
- Choose **TDML File (*.tdml, *.tdm)** from the **Save as type** drop-down list box.
- Type in a file name with either a **.tdml** or **.tdm** file extension. The 3-letter extension currently has the most support because it is compatible with CD-ROMs and older Windows system.
- Press **OK** to save the file.

9.9 Printing the Diagram Window

To print the Diagram window:

- Choose the **File > Print Diagram** menu option in the timing window. This will open the *Print* dialog (see Figure 9.1 on the following page).
- Set the printing options and press the **Print** button. The following list explains the printing options for the Drawing window. Note: Press the **Apply** button to save the printer settings without actual performing a print operation (cancel to close the dialog).

.



Print

Printer
Lexmark Optra R plus

Print Time Range
( ) Entire Time Scale
(●) Time From: 0 To: 350

Print Signal Range
(●) All Signals
( ) Currently Selected Signals
( ) Currently Displayed Signals

Print or Create Image File
Print to: Printer
Filename:
☐ Preview: (●) Interchange ( ) TIFF 5
Preview Resolution (DPI): 150

Margins
☑ Use Margins
Margin sizes in inches:
Sides: .75 Top/Bot: 1.00
Custom header and footer strings:
Header: %d %t;%f;%p
Footer:

Objects to Print
☑ Print TimeLine ☑ Print signal names
☑ Print border box ☑ on each page

Scaling (10 - 200%)
% Horizontal: 100 % Vertical: 100
Number of horizontal pages: 1

Figure 9.1 Print Dialog

The following list explains the printing options for the Diagram window:

1) Printer:
Displays the default printer. Use the **Setup** button to change the default printer.

2) Print Time Range:
Entire Time Scale: Prints to the end of the longest signal (not including clocks).
Time: Prints from the starting time to the ending time.

3) Print Signal Range:
All signals: Prints all signals that are not currently hidden.
Currently Selected Signals: Prints only signals that are highlighted.
Currently Displayed Signals: Prints only signals whose names are completely displayed in the Diagram window.

4) Print or Create Image File: (image.doc has more information)
Print to: Lists the printer and image options:
Printer: Prints to the default printer. If active, all other options in this section do not apply.
File: Prints to a file in a format that is compatible with the default printer. A file name must be provided in the **file name** edit box.

WMF Metafile [MS Word]: Copies a metafile to the clipboard and optionally saves a disk-based metafile.

MIF FrameMaker: Prints to a Maker Interchange File (MIF) that can be edited in the FrameMaker desktop-publishing program. A file name must be provided in the **file name** edit box. Once the MIF radio button is checked you can optionally set the horizontal size of the image using the check box and edit box in the Margins section of the dialog. Setting the image size is useful when you want to generate multiple MIF images of the same size. MIF files are scalable once they are imported into FrameMaker.

CGM Metafile: Prints to a CGM metafile. Enter a **file name** with a **.cgm** extension. There is some cross-platform support for CGM metafiles so you can test compatibility with your MAC and UNIX programs.

TIFF Tagged Image File Format: Prints to a TIFF file. Enter a filename with a **.tif** extension. TIFF files have a very high quality loss-less data compression.

PNG (DataSheet Pro only): Prints to a PNG file. Enter a filename with a **.png** extension. This image format is web-ready with a high quality loss-less data compression.

JPEG (DataSheet Pro only): Prints to a JPEG file. Enter a file extension of **.jpg** for your filename. This web-ready format has a lossy data compression technique that ensures minimal size for your graphics.

EPS Encapsulated Postscript File: Prints to a cross-platform, encapsulated postscript file (EPS). The Preview check box works with the EPS File option to create an EPS file that can be displayed inside of other programs like Word and FrameMaker. Without a preview, imported EPS files generally show up as a gray or white box in most applications. Two different preview formats are available: **Tiff 5** which is compatible with Microsoft Office applications, and **Interchange** which is compatible with FrameMaker. Previews make an EPS picture viewable inside an application, but the trade-off is an increase in the size of the image. Setting the Dots Per Inch (DPI) edit field controls the resolution and size of the preview image. For more information on EPS support please view the image.doc file included on the distribution disk.

WARNING: an error in Word 7.0 causes previews to be printed even when printing to a Postscript printer. Contact Microsoft to get a free update to Word 7.0a that fixes this problem.

5) Margins (non FrameMaker)

Use Margins: Enables the margin options. If this is unchecked the printing will extend as far left and right as your printer will allow. This option combined with an unchecked **Print signal names** is useful if you are trying to tape long diagrams together.

Sizes in inches: Indicates how wide the margins should be.

Custom header and footer strings: Headers and footers can be placed on a diagram printout using header and footer strings. Control codes can be embedded into the header and footer strings. Below is a list of the available control codes:

%f	filename
%p	page number
%d	date
%t	time

Text can be left, right, and center justified using a semicolon ';' to separate the text strings. **For example:** "Filename %f ;;%p" would left justify the filename, put no text in the center, and right justify the page number.

Margins (for FrameMaker print setting):

If the **Print To: MIF** radio button is selected then the **Margins** section of the dialog has different controls that can be used to set the horizontal MIF image size.

Specify Image Size: Enables the width edit box.

Width: accepts the width of the MIF image in inches.

6) Image Views:

Image View box: Double clicking on a view loads the view information into the *Print* dialog. Read section *Managing Multiple Views (Images) of a Diagram* for more information

Add: Saves current printer settings in a view.

Delete: Removes and deletes the selected view.

7) Objects to Print

Print TimeLine: If checked, the timeline is printed.

Print border box: If checked, a black border is printed around the drawing.

Print signal names: If checked then the signal names will be printed.

on each page: If signals are longer than one page width, this box indicates whether or not the signal names will be printed on the overflow pages. "Print signal names" must be checked in order for this to affect the printing.

8) Scaling

% Horizontal and % Vertical: Indicate the amount to scale the drawing. Normal size is 100%, smaller pictures are less than 100%, and larger pictures are greater than 100%. We recommend you use TrueType fonts when scaling, as regular fonts do not scale.

Number of horizontal pages: Forces the drawing to be printed on the specified number of horizontal pages. This feature only scales the waveforms horizontally. Text and signal names remain the same size.

9.10 Printing the Parameter Window

To print all the parameters in the parameter window:

- Select the **File > Print Parameters** menu option to open the *Print Parameters* dialog box.

- Set the printing options and press the **Print** button. The following list explains the printing options for the Drawing window. Note: Press the **Apply** button to save the printer settings without actual performing a print operation (cancel to close the dialog).

The following is a list of parameter window printing options:

Print Display:

Displays the default printer. Use the **Setup** button to change the default printer.

Print or Create Image File:

- **Printer to:** Lists the printer and image options:
  - **Printer:** Prints to the default printer. If active, all the other options in this section do not apply.
  - **File:** Prints to a file in a format that is compatible with the default printer. A file name must be provided in the **file name** edit box.
  - **All other image file formats:** These are covered in the *image file* and *Print the Diagram* sections of this chapter.

Image Views:

- **Image View box:** Double clicking on a view loads the view information into the *Print* dialog. Read section *Managing Multiple Views (Images) of a Diagram* for more information
- **Add:** Saves current printer settings in a view.
- **Delete:** Removes and deletes the selected view.

Note: The parameters will be printed in the current parameter window font. TrueType fonts work best when printing.

Chapter 10: Parameter Libraries

Parameter libraries are free parameter files that can be shared by multiple timing diagram projects. Libraries contain the timing parameter information of components. The timing diagram editor ships with several standard libraries that contain over 10,000 timing parameters, and it also supports the industry standard TDML for on-line component information. You can also use the timing diagram editor to create your own libraries.

Using Parameter Libraries: Add a parameter library to the project, make a specification, and define a macro. Once the Parameter Library is set up you can reference the library parameters in your project.

10.1 Adding Parameter Libraries, Specifications, and Macros
10.2 Using Library Parameters (Referencing)

For more information on making, converting, accessing, and maintaining libraries read the following sections:

10.3 Making Parameter Libraries
10.4 Converting Parameter Libraries

Maintaining Parameter Library and project integrity: Parameter libraries and projects change, so there are several features for updating, editing, browsing, and archiving your projects and libraries.

10.5 Merging a Parameter Library into a Project File
10.6 Copying Referenced Parameters
10.7 Editing Parameter Libraries
10.8 Updating a Project from the Parameter Libraries
10.9 View Parameter Library Parts

10.1 Adding Parameter Libraries, Specifications, and Macros

Before you can use a parameter library in a timing diagram project you must add the library to the library search list, set up a specification, and define a macro for the library. Here are the three basic steps:

1) Add Parameter Libraries to your Library Search List:

- Select the **Libraries > Parameter Library Preferences** menu option to open the *Parameter Library Preferences* dialog.
- Press the **Edit Parameter Libraries** radio button. If you are creating several timing diagram project files and you want all of them to use the same default library list, then choose the **Edit Default Libraries** radio button.

- Click the **Add Library to List** button on the right side of the dialog. This opens the *Library Browse* dialog.
    - Select the names of the parameter libraries that you want to add to the library list.
    - Click **OK** to return to the *Library Preferences* dialog.
    - Notice that the **Use full pathnames** checkbox is checked. This means that both the library filename and the path information will be added to the library list. If you wish to reference libraries contained only in the current working directory then you can uncheck this checkbox. Usually libraries are organized in a tree directory structure with different paths for different manufacturers, so you will probably want to leave this checked.
- The *Library Preferences* dialog now contains the libraries you selected are listed in the **Current library list**. At this point you can reference and use the values in the libraries. DO NOT close this dialog as it is used in the next section (this dialog is modeless, so you can leave it open while you perform regular operations).

2) Define Specifications for the Parameter Libraries (Optional, but recommended):

- **Theory:** When you use a parameter name in a formula, the program searches the project for the definition. If the definition of the parameter is not found in the project, the libraries are searched in the order they are listed in the Library search list until a match is found. If no match is found, the formula will display the error symbol. Since different libraries may have parameters with the same name, you need to set up a way to differentiate which library should be searched to find a particular parameter. Library Specifications let you differentiate which library to search.

  Also, Parameter Libraries with library specifications will only be searched if the parameter being sought has a matching specification. Therefore, if you keep a large number of libraries on your library search list, it is a good idea to give them library specifiers so that parameter searches are quick.
- Inside the *Library Preferences* dialog, notice the **Specification** box to the right of the Current library list box.
- Select a library to modify its specification.
- Click the right arrow --> once to add a specification. Notice that the name of the library has moved into the specification list. Each time you click the right arrow, more of the path is added to the specification. Usually the filename is a sufficient specification.
- Now when you want to reference a parameter in a specific library, use this notation:

    LibrarySpecification:ParameterName
- Click **OK** to close the dialog.

The next section, *10.2 Referencing Library Parameters*, shows how to reference a library parameter without having to type the name and specification.

3) Define a Macro to represent the specifications of the new Parameter Library (Optional):

- **Theory:** Macros allow you to substitute one string or character for another in formula evaluation. If you make macros for your library specifications and use the macros in your parameter formulas, you can analyze the impact of switching to parts from another logic family or another vendor by changing the macro values to other library specifications (assuming the parameter names are the same across libraries). You only need to use macros if you plan to switch between libraries.
- **Example:** Macro Name => %lib%   Value => motorola:als:

  A parameter value of **%lib%partname** will equate to **motorola:als:partname**. Now you could define **%lib%** to be **ti:ls:** and the example will evaluate to **ti:ls:partname**. Assuming that parameter names remain the same between libraries and the appropriate libraries are on the current library list, you can use this technique to switch manufacturers and logic families.
- Select the **Libraries > Macro Substitution List** menu option to open the *Edit Macro* dialog.
- Type the macro name into the **Name** box. *No percentage characters are allowed in macro names.*
- Select the macro value from the **Value** drop down box.
- Click the **OK** button to add the macro or press the **<Enter>** key to add the macro and close the dialog box.

10.2 Using Library Parameters (Referencing)

Referencing library parameters in the formulas of other parameters is a way to use libraries without copying the library parameters into the project. This is particularly useful when you are sharing libraries between several projects and you don't want to manually update each project if a library changes.

To reference a library parameter use the following syntax:

LibrarySpecification:ParameterName

For example, if the library specification is **ti:als** and the parameter name is **00;tpLH**, then to reference that parameter you would use **ti:als:00;tpLH**. If the library has no specification, you can reference parameters from it by just typing the parameter name (i.e. 00;tpLH). To avoid excessive keystrokes and possible syntax errors, here are some techniques for automatically adding a library parameter to your formula:

- Double click on a delay, setup, or hold in your timing diagram to open the *Parameter Properties* dialog.
- Click the **Libraries** button in the bottom-right side of the dialog to open the *View Parameters in Library* dialog.
    - Select a Library to view its parts. Notice that the list box to the right is filled with timing parameters.
    - Select a timing parameter to insert into your formula. Scroll around until you find the parameter, or type a portion of the name into the **Search For** edit box.
    - Choose one of following three buttons to copy the library parameter into the project:
        1) Click the **Insert into Formula** button, which will copy the *+LibrarySpecification***:***ParameterName* into both the min and max edit boxes of the parameter. The unary plus operator makes it easy to copy the parameter into a partially complete formula. If you are just referencing the parameter value then the unary plus does not affect the value. This is the most common method to reference the library parameter.
        2) Click the **Replace Parameter** button. This overwrites the name, min, max, and comment values of the parameter displayed in the *Parameter Properties* dialog.
        3) Click the **Insert into Table** button. This creates a new free parameter in the Parameter Table which has the same name, min, max, and comment values as the library parameter. No changes are made to parameters displayed in the *Parameter Properties* dialog.
    - Click **OK** to close the dialog and return to the *Parameter Properties* dialog box.

    Note: If a parameter is selected, the **OK** button performs the same function as **Insert into Formula.**
- Notice that both the min and max edit boxes are filled with the parameter values.

Alternate Method:

- Inside the *Parameters Properties* dialog, click inside the **min** or **max** edit box.
- Press the **F3** key on the keyboard, to open the Library Parameters dialog.
- Select a parameter just like you did in the first method and click **OK** to close the dialog and return to the *Parameters Properties* dialog.
- Notice that parameter value was copied into only one of the min or max edit boxes.

10.3 Making Parameter Libraries

Parameter libraries are files that store parameters. These libraries can be stored in three different file formats:

1) **A Tab or Comma Separated format (*.txt):** This file format is the fastest loading format and is compatible with spreadsheet programs.

2) **Free parameter files (*.fp):** The SynaptiCAD free parameter format. The base time unit is stored with the library file so the file can be used with a project that has a different base time unit than the library.

3) **A Timing Project file (*.tim):** A complete timing diagram can be used as a library. However, only the parameters will be read from the file. This is the slowest loading format, but it is handy when you want to quickly reference a parameter from another project.

Parameter libraries can be made using the timing diagram editor or they can be imported using a commercial spreadsheet. You will probably want to develop large libraries using a commercial spreadsheet because they have more viewing and editing features than the built-in Parameter table.

To make a parameter library using the timing diagram editor:

- Open an existing project that has free parameters in it, or add and edit free parameters using the parameter table.
- Select either **Libraries > Save Free Parameters As Library** or **Libraries > Save All Parameters As Library** menu option depending on which parameters you want to save to the file. This will open the *Save As* dialog box.
- In the **Save as type** drop-down list box choose either **Free Param Text (*.txt)** for the tab separated format, or **Free Parameter (*.fp)** for the free parameter file format. The **txt** format is recommended.
- Choose a file name and then press the **Save** button to create the library file.

To make a parameter library using a commercial spreadsheet:

- Open your spreadsheet program.
- Type **NAME**, **MIN**, **MAX**, and **COMMENT** into each column of the first row of the spreadsheet.
- On each subsequent row, type the values for a single parameter
- Use the **Export** or **Save As** menu command to create a Tab-separated text file (or consult your spreadsheet documentation for details). **Note:** Test small files for compatibility before beginning a large library project.

10.4 Converting Parameter Library File Formats

Tab Separated (txt) library files can be converted to the free parameter format. The only reason that you may want to do this is because the current Base Time Unit and Display Time Unit are only stored with the library data in the free parameter format. Because txt files do not store this time unit information, users need to be careful that the library data is used properly (i.e. ensuring that the current

Display Time Unit matches the intended time unit of the library). However, text files load significantly faster so you may want to keep your parameter libraries in this format.

To convert a comma/tab separated library into free parameter format:

- Select the **File > New Timing Diagram...** menu option to make sure that no project is currently loaded.
- Set the **Options > Display Time Unit** to match that of the library you are about to import.
- Use **File > Open Timing Diagram...** menu item to open the file that you want to convert.
- Look at the Parameter Table Window and verify that the parameter values are correct.
- Next, save the library by using the **Libraries > Save Free Parameters As Library** menu to open the *Save As* dialog.
- Choose **Free Parameter (*.fp)** for the free parameter file format from the **Save as type** drop-down list box. Save the file using the **.fp** extension.

10.5 Merging a Parameter Library into a Project File

When a library is merged into a project, all the free parameters in the library are copied into the project. During the merge process, library free parameters take precedence over project free parameters of the same name (the library data overwrites the project data if there is a conflict).

To merge a library into a project:

- Open the project file (.tim).
- Choose the **File > Merge Timing Diagram...** menu option to open the *File* dialog.
- Type in the name of the library file and choose **OK**.

Note: Merging should generally only be used with small libraries as you probably don't want all the parameters cluttering up your work space in the parameter window. To clean up your work space, parameters in the parameter window can be hidden by selecting the parameter and choosing the **View > Hide Selected Parameter Data** menu option in the parameter window.

10.6 Copying Referenced Parameters

During the design, process projects usually reference free parameters contained in parameter libraries. But when the design is finished and it is time to prepare the project files for documentation and archiving, it is inconvenient to have to carry around libraries with your project (and you may not want the parameters to change if the libraries are updated). In this case it is generally preferable to copy all free parameters used into the project so that the project (.tim) file contains all the free parameters it needs to draw itself.

To make a self-contained project:

- Choose the **Libraries > Copy Referenced Library Parameters into Table** menu option. This copies into the project every library free parameter that is used by the project.
- A message will appear asking you if you want to hide the free parameters when you copy them into the project.

If you are planning to move your project, then you should empty the parameter library list so that the program will not attempt to read library files not found at the new location. If you are not moving your project then leaving the library list intact won't affect your project because parameters in the project take precedence over those in the libraries.

10.7 Editing Parameter Libraries

To edit a parameter library:

- Load the library. Choose **File > Open** menu and type the name of the library.
- In the parameter window, find and update the free parameters as desired.
- Choose **File > Save Free Parameters** menu option in the parameter window to save the library file.

Any projects that reference the library will now have access to the updated free parameter, unless the parameter has been copied into the project.

If you have made a local copy of the parameters in your project, you can update them by the following procedure:

- Load the project into the timing diagram editor.
- Choose **Libraries > Update Parameter Table from Parameter Libraries** menu option.

10.8 Updating a Project from the Parameter Library

Normally free parameters inside a project take precedence over free parameters in libraries specified in the Library List. Sometimes it is necessary to update free parameters in a project with the data contained in a library.

To update a project with data from *every library* in the library list:

- Select the **Libraries > Update Parameter Table from Parameter Libraries** menu option. This replaces the timing data of free parameters in the project with data from library free parameters of the same name.

To update a specific free parameter with data from a *certain library*:

- Use the **Libraries > View Parameter Library Parts** menu option and the **Insert into Table** button. See the instructions in the *View Parameter Library Parts* dialog topic.

10.9 Viewing Library Parts

The *View Parameter Library Parts* dialog displays the free parameters contained in the libraries specified on the library search list. It also lets the user selectively copy parameters into the parameter table used by the project.

To view parts in a specific library:

- Select the **Libraries > View Parameter Library Parts** menu option. This will either open the dialog or inform you that you need to first put the libraries on the library list using the *Parameter Library Preferences* dialog.
- Select the library that you would like to view from the library list box. The library parameter names will be displayed in the *Parts List* box.

To copy library free parameters into the project:

- View the library parts according to the above directions.
- Select the parts that you want to copy into the project.
- Click on the **Insert into Table** button. The parameters will be copied into the parameter window, with the library's specification, if any, prepended to them.

Chapter 11: Waveform Equation Generation

WaveFormer Pro, VeriLogger Pro and TestBencher Pro have the ability to generate signal waveforms from Temporal equations and automatically label those waveforms using State Label equations. These features augment the drawing environment, and provide a quick way to generate signals without having to draw each individual signal transition. They also provide a way to generate signals that have precise edge placement.

Waveforms can also be generated using registered Boolean equations simulated by the Interactive HDL Simulation engine inside WaveFormer, VeriLogger Pro, and TestBencher Pro. Chapter 12 covers the Interactive HDL Simulation features.

Background: The temporal equation and state label equation features are implemented using dynamic Waveperl scripts. Users can write their own dynamic Waveperl scripts to extend the functionality of WaveFormer, VeriLogger and TestBencher Pro. For more information read *Chapter 14: Writing Waveperl Scripts*.

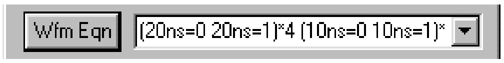
11.1 Generating Waveforms from Temporal Equations

Temporal equations provide a quick way to generate signals that have a known pattern that is more complicated than a periodic clock.

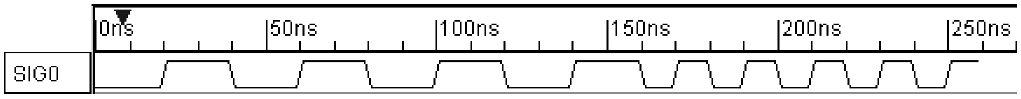
Many signals are easier to create using a temporal equation instead of drawing them. For example, a change in frequency in a periodic waveform from 25Mhz to 50Mhz could be represented by the following temporal equation:

(20ns=0 20ns=1)*4 (10ns=0 10ns=1)*5

Temporal equations are entered in the *Signal Properties* dialog.



(20ns=0 20ns=1)*4 (10ns=0 10ns=1)*



This creates a signal with an initial frequency of 25MHz (period = 20+20 = 40ns) for 4 cycles and switches to 50Mhz for 5 more cycles. This type of waveform is tedious to draw by hand, but can be concisely expressed as a temporal equation.

To Apply a Temporal Equation to a signal:

- Double click on a signal name to open the *Signal Properties* dialog.

- Enter the temporal equation into the edit box next to the **Wfm Eqn** button.

Note: The default equation contains all of the available states and syntax. If you start by editing the default equation then you do not have to memorize the temporal equation rules.

- Press the **Wfm Eqn** button to apply the equation. Each time you press this button, the waveform will be appended to the end of the signal.

Temporal equations must follow certain syntax rules:

- The equation is a list of space-separated, time-value pairs in the form:

  TimeValue[units]=StateValue. For example: **10ns=Z** is a valid time-value pair.

- If **[units]** are not specified, then the "display time unit" of the current project will be used.

- Legal **StateValues** are: 0 = strong low, 1= strong high, Z = tristate, V = valid, X = invalid, L = weak low, and H = weak high.

- For loops, enclose a list of time-value pairs in parenthesis and use the multiply symbol '*' followed by the number of times the list is to be repeated.

11.2 Label Waveforms using State Label Equations

State Label equations provide an easy way to label signals with state patterns. For example, your design may contain a counter and you would like to display the actual count values, instead of empty valid segments. Once your counter signal is drawn you can write an equation to label the segments.

A simple counter that starts at one and counts to ten is written as:

Inc(1,1,10)

This tells the program to start at one and place ten labels, incrementing by one each time. The list of all usable functions is given below.

To Apply a State Label Equation to a signal:

- Double click on a signal name to open the *Signal Properties* dialog.

- Enter the State Label equation into the edit box next to the **Label Eqn** button.

Note: The default equations contain examples of the state label functions. If you start by editing a default equation you may not have to look up the function definitions.

- Press the **Label Eqn** button to apply the equation.

Note: State Label equations are only calculated once, and any values past the end of the signal are discarded. The old virtual state values inside segments will be overwritten by the State Label Equation.

Tip: It is often useful to use the **Temporal Equation** feature to create an unlabeled signal with the correct number of segments, and then label it with a State Label Equation.

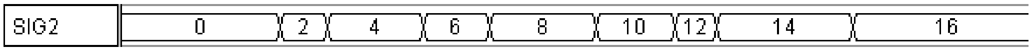
State Label Equation syntax rules:

State Label equations are a made up by combining the various label functions. Functions can be applied in sequence by listing them in a comma separated list. Also, each label function returns a value of **list**. Whenever you see **list,** you can use a function, a single value like "5" or "blue", or a list in parentheses (2,3,4). Below is the list of functions:

Inc(start, increment, count)

This function will generate labels that begin at "start" and increase by "increment" each time. A total of "count" labels will be generated.

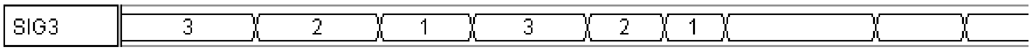
Example: **Inc(0,2,10) -** Labels the signal with the values 0,2,4,6,8,10,12,14,16,18



Dec(start, increment, count)

This function generates labels that start at **start** and decrease by **increment** each time. It is equivalent to calling Inc(start, -increment, count)

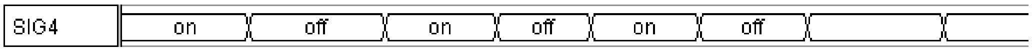
Example: Rep(**Dec(3,1,3)**,2) - Labels the signal with the values 3,2,1,3,2,1



Rep(list, count)

This function repeats **list, count** number of times. The "list" can either be a literal list (e.g. (1,3,4) or ("red","green","blue") or it can be a function that generates a list (e.g. Inc, Dec, or Rep)

Example: **Rep(("on", "off"), 3) -** Labels the signal with the values on, off, on, off, on, off. If the list contains words, place it in quotes.

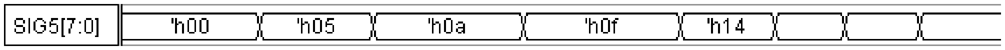


Hex(list)

This function formats the list as hexadecimal numbers. All numbers in the list are converted to a base-16 representation. The letters a-f are lowercase, and the numbers are prefixed by "**'h**" to indicate that they are hexadecimal.

By default, all numbers in the list are formatted to the same number of digits, adding leading zeros as necessary. If a MSB and LSB for the signal are specified in the *Signal Properties* dialog, all numbers will be formatted to display the given number of bits.

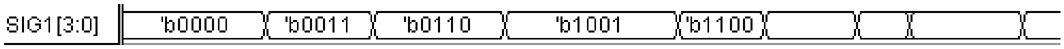
Example: **Hex(Inc(0,5,5)) -** Labels the signal with the values ’h0, ’h5, ’ha, ’hf, ’h14



Bin(list)

This function formats lists as binary. By default, all numbers in the list are the same length, with leading zeros as necessary. If a MSB and LSB for the signal are specified in the *Signal Properties* dialog, all numbers will be formatted to display the given number of bits.

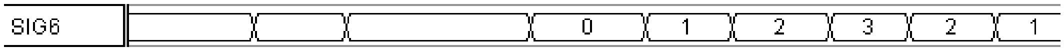
Example: **Bin(Inc(0,3,5)) -** Labels the signal with the values 0000,0011,0110,1001,1100



Skip(Integer)

This function format causes a number of segments to be skipped (left unchanged) by the equation.

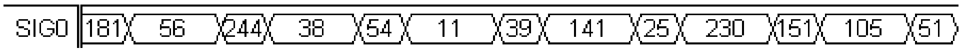
Example: **Skip(3), Inc(0,1,4), Dec(2,1,3) -** Labels the signal with the values 0,1,2,3,2,1,0, starting at the fourth segment. The first three segments are skipped, leaving them with their original label (blank if they had no label to start with). Any number of functions can be chained together in this style, separating them with commas.



RandomIntegerArray(count, Range_to_zero) or **RandInt (count, Range_tozero)**

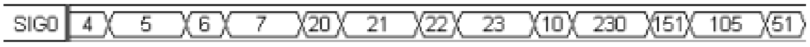
This function generates random state values for waveforms. Count indicates the number of integers to generate. The integers are random values within an inclusive range of 0 to the Range value.

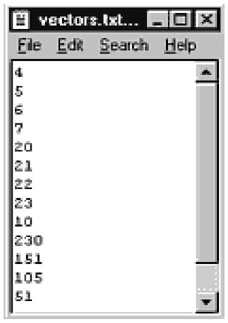
Example: **RandomIntegerArray(8,255) -** Labels 8 waveform segments with random integers within a range of 0 to 255 (inclusive).



File(’*filename.txt*’)

This function imports a set of state values from the specified file. This is an easy way to import values generated in a program such as MatLab, Mathematica, or a user-written C program. In the file, each state value should be on a separate line.



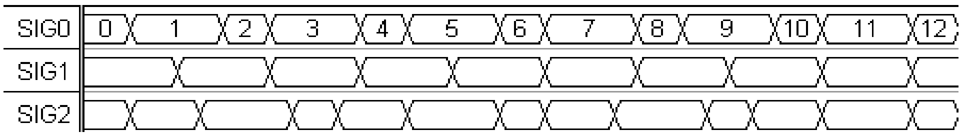


```
4
5
6
7
20
21
22
23
10
230
151
105
51
```

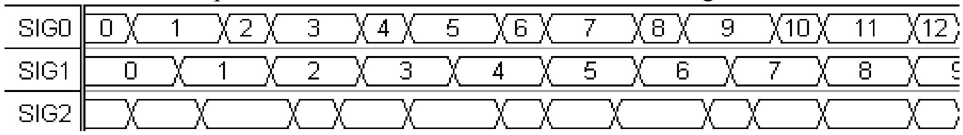
The figure to the right shows the a series of values stored in a text file; the figure above shows the state values after the File label equation has been applied. The values are read in from the file and applied to the states of the waveform sequentially.

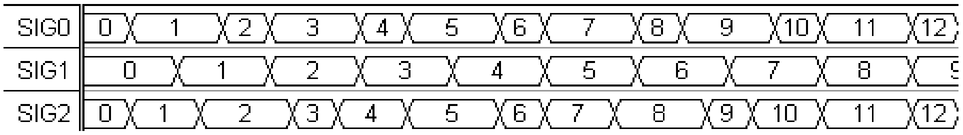
Signal("*signalname*")

This function copies the states from the signal indicated by *signalname* and applies them to the signal being operated on. If no *signalname* is specified then the function returns the array of states of the current signal (this is useful when working with the **map** function).



Example: **Signal("SIG0")**. The figure shown above would be a starting diagram, with the second two signals not having state values assigned. Applying the Signal function to SIG1 produces the values shown in the next figure; likelwise, applying the Signal function to SIG2 produces the state values shown in the third figure.

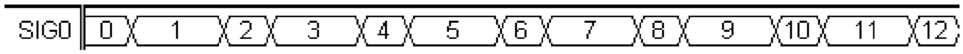




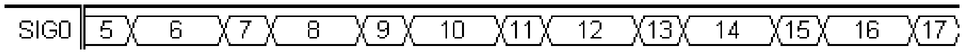
map{*operations*} *array*

The map function allows you to perform a set of operations on every state in a state array. The first argument to the map function is the set of operations to perform, enclosed in brackets. The second argument is the array whose elements are to be operated on. The special variable $_ is used to represent the state value to operate on (it takes on the value of each state in the array). Once you get familiar with it, the map function in combination with standard Perl language operations gives you extraordinary expressive power.

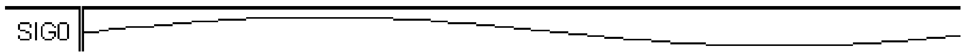
Example: **map { $_ + 5 } Signal() -** The Signal function returns an array of the states on the current signal, and the map function increments each state in the array by 5.



The figure above shows the states before the map function is applied; the figure below shows the resulting states.



Example: **map { sin($_) } Range(0,2*3.14159,100) -** The Range function generates an array of 100 values between 0 and 2 PI radians. The map function computes the sine value for each element in this array. This technique is useful for generating analog waveforms. Below is an equation that generates a set of values that can be displayed as a sine wave using the analog display capability.



Chapter 12: Interactive HDL Simulation

WaveFormer Pro, VeriLogger Pro and TestBencher Pro have a built-in Interactive HDL Simulator that can simulate Boolean Equations with delays, register and latched signals, and behavioral Verilog code. The simulator is interactive, which means modifications to input waveforms will trigger instant resimulations. This feature greatly reduces the amount of time needed to draw a timing diagram, especially one that models gate level circuits.

All equations are entered through the Logic Wizard section of the *Signal Properties* dialog, which is reached by double clicking on a signal name.

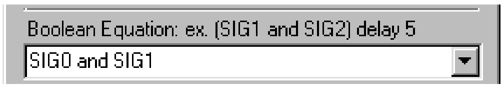
12.1 Generating Waveforms from Boolean Equations

In WaveFormer, VeriLogger and TestBencher Pro, Boolean equations relate one signal to other signals in the diagram. To describe a signal with a Boolean equation:

- Double left click on a signal name to open the *Signal Properties* dialog.

- Make sure the **Boolean Equation** radio button is selected.

- Enter a Boolean equation into the edit box.


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

SIG0 and SIG1

- Push the **Simulate Once** button and watch the signal draw itself. If the **Simulate** radio button is selected, any changes to the input waveforms will cause a resimulation.

- To view or edit the Verilog HDL code used by the simulator, select the **HDL Code** radio button.

The Boolean Equation edit box accepts Boolean equations in VHDL, Verilog, and SynaptiCAD's enhanced equation syntax. The SynaptiCAD format supports the operators: **and**, **or**, **nand**, **nor**, **xor**, **not**, and **delay**. The delay operator takes a signal on the left and a time or parameter name on the right and returns a signal. If a parameter name is used on the right hand side of the delay operator, then the equation will simulate true min/max timing. This true min/max timing analysis is the main advantage that SynaptiCAD's format has over the VHDL or Verilog format. Instead of min/max timing, Min-Only or Max-Only simulations can be performed by changing the **Options > Simulation Preferences > Timing Model** drop-down list box. Below are sample Boolean equations:

(SIG0 and SIG1 and SIG3) delay 20ns

This models a 3-input AND gate with a 20ns delay.

(SIG0 delay 20ns) and (SIG1 delay 10ns)

This models an AND gate with 2 different input delays.

(SIG0 and SIG1) delay GateDelay

Assume GateDelay is a delay parameter with a min time of 15ns and a max time of 20ns. This models an AND gate with a delay between 15ns and 20ns. Each edge of the simulated signal will have a grey uncertainty region of 5ns.

IMPORTANT: When a Boolean equation is resimulated the signal is deleted and recreated as a new signal. The program attempts to preserve the attachment of parameters and text objects on signal edges, but the addition or loss of edges during the Boolean computation may move these objects.

12.2 Advanced Gate Representation

The **Boolean equation** edit box of the *Signal Properties* dialog can accept several advanced operators, like conditional expressions and signal concatenation. These operators can be used to model multiplexers, tristate gates, and multi-bit signals. The following demonstrates several of these modeling techniques:

Conditional Expressions for Multiplexers and Tristate gates: The normal C language conditional expression of ***conditional ? if_expr : else_expr*** can be used inside the Boolean Equation edit box to model multiplexers and tristate gates. Some examples are:

- Tristate Gate: **EnableSig ? SIG0 : 'bz**
- 2-1 MUX: **S0 ? SIG0 : SIG1**
- 4-1 MUX: **S1 ? (S0 ? SIG0 : SIG1) : (S0 ? SIG3 : SIG2)**

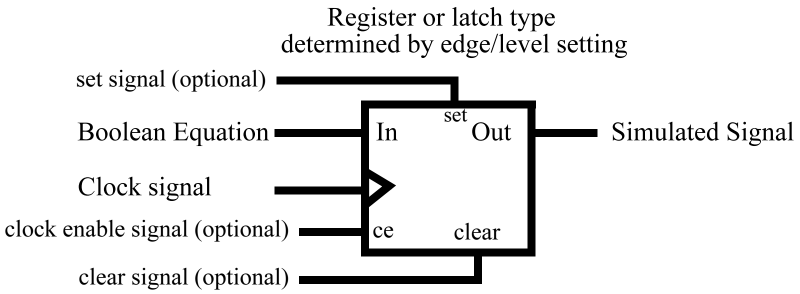
Multi-bit Equations are specified by setting the **MSB** and **LSB** of the signal (located at the bottom of the *Signal Properties* dialog). To change a simple 1-bit equation to a 4-bit equation, all you have to do is set the MSB of the signals involved to 3.

Concatenation of Signals is supported using the Verilog concatenation operator. You must set the **MSB** in the *Signals Properties* dialog to the proper size. If the size of the concatenated signal is larger than the receiving signal, then the most significant bits are dropped. Some examples of the concatenation operator:

- Signal Concatenation: **{SIG0, SIG1}**
- Concatenating bit-slices: **{SIG0[3:0], SIG1[7:4]}**

12.3 Generating Waveforms from Register and Latch Equations

WaveFormer Pro, VeriLogger and TestBencher Pro support simulation of registered and latched equations. The logic wizard is used to enter information about the circuit.



To describe a signal that is registered or latched:

- Double left click on a signal name to open the *Signal Properties* dialog.
- Make sure the **Boolean Equation** radio button is selected.
- Enter the input signal name into the Boolean equation edit box. The input Boolean equation can either be the name of the input signal or an equation that conditions the input signal.
- Choose the clocking signal from the **Clock** drop-down list box. The clocking signal can be any clock or signal in the timing diagram.
- Choose the type of edge or level triggering from the **Edge/Level** list box. For a Register circuit choose **neg** for negative edge triggering, **pos** for positive edge triggering, or **both** for edge triggering. For a Latch circuit choose either **low** or **high** level latching.
- The **Set, Clear,** and **Clock Enable** are optional signals that model the set, clear, and clock enable lines of the register or latch. If "Not Used" is chosen for a line, then that line is not modeled. These lines can be active low or high and synchronous or asynchronous depending on the settings in the *Advanced Register and Latch Controls* dialog (see next bullet).
- The **Advanced Register** button opens the *Advanced Register and Latch Controls* dialog which determine how this individual register is generated. The global defaults can be defined using the **Options > Simulation Preferences** menu. This dialog controls the following options:
    - **Clock to Out:** Describes the delay from the triggering of the clock signal to a change on the output edge.
    - **Setup:** Describes the time for which the input must be stable before the clock-triggering event. If a min/max time pair is entered, Setup will use the min time. Any violations of this setup time will be reported to the simulation log file **verilog.log**.

- **Hold:** Describes the time for which the input must remain stable after the clock-triggering event. If a min/max time pair is entered, Hold will use the min time. Any violations of this hold time will be reported to the simulation log file verilog.log.

In the **Clock Enable** area:

- **Active Low:** If checked, the clock will be enabled when the *clock enable* line is low. If unchecked, the clock will be enabled when the *clock enable* line is high.

In the **Set and Clear** area:

- **Active Low:** If checked, the set and clear lines will control the output when they are low. If unchecked, then the set and clear lines will control the output when they are high.
- **Asynchronous:** If checked, then the set and clear lines will control the output anytime they are active. If unchecked, the model is synchronous and an active set or clear line does not affect the output until the next clock trigger event.

- Push the **Simulate Once** button to trigger a simulation. If the **Simulate** radio button is selected any change to the input waveforms will cause a resimulation.
- To view or edit the Verilog HDL code that is used by the simulator, select the **HDL Code** radio button.

12.4 Multi-Bit Simulation

WaveFormer Pro, VeriLogger and TestBencher Pro support multi-bit signals for both simulation and stimulus generation. Multi-bit signals can display their signal values as binary, hexadecimal, or decimal numbers.

To define a multi-bit signal:

- Double click on the signal name to open the *Signal Properties* dialog.
- Set the bits directly in the signal name by bracketing the MSB and LSB information (e.g. ADDR[31:0] sets MSB to 31 and LSB to 0).
- OR enter the values into the **MSB** and **LSB** edit boxes. When the new MSB and LSB values are applied, the name of the signal will change to display the width of the signal.

Note: The width of the signal is not a part of the signal name. For example: in equations, the 32-bit signal ADDR[31:0] is still referenced as ADDR. The "[31:0]" is not a part of the signal name.

To insert the values into the segments of a multi-bit signal:

- Choose the base of the number system to display the signal values by setting the **Radix** drop-down list box in the *Signal Properties* dialog. Currently, three bases are supported: binary (bin), decimal (dec), and hexadecimal (hex).

- Draw the signal waveform using Valid segment states.
- Double-click on a **segment** in the signal to open the *Edit Bus State* dialog.
- Type in an appropriate value into the **Virtual** edit box.
- Use the **Next** and **Prev** buttons to edit the rest of the signal.

Note: If a signal has a radix of hex or decimal, and one of the bits of that signal is in an unknown state (X), then that segment will be displayed as a binary value. For example if a segment on SIG0 had a value of 10X1 and a radix of HEX, then the value displayed would be 'b10X1 since that value is not a valid HEX number.

12.5 Direct HDL Code Simulation

WaveFormer provides the user with the ability to directly enter HDL code for simulating signals that are not easily modeled using simple Boolean equations (e.g. you want to write behavioral HDL code). To enter direct HDL code instead of a Boolean Equation:

- Double left click on a signal name to open the *Signal Properties* dialog.
- Select the **HDL Code** radio button. This radio button changes the *Signal Properties* dialog so that a multi-line edit box is displayed in place of the regular Logic Wizard controls.

There are several important points to remember when writing direct HDL code inside a signal's HDL Code Editor:

1) **All the HDL code is placed inside a module.** This places one important restriction on the HDL code: it *cannot* contain an HDL module declaration (i.e. you cannot define a new type of component inside a signal's direct HDL code block). You CAN create an instance of an HDL module, however, as long as the module is declared somewhere else (e.g. in wavelib.v). The advanced simulation tutorial covers this in more detail.

2) **WaveFormer signals are always treated as wires during HDL code generation.** This makes it easy to write simple continuous assignment statements. This also means that if you want to control the signal state using behavioral code, you will need to create a **reg** as a intermediate variable and then assign the SynaptiCAD signal to the value of the intermediate reg. This is because the values of wire (nets) cannot be set inside a behavioral (initial or always) block. For example, direct behavioral code for a signal called SIG0 that inverts whenever CLK changes state, you would need to write:

```
reg tempSIG0;                //temporary reg
always (@CLK)
  begin
  tempSIG0 = !tempSIG0;      //invert temporary whenever CLK changes
  end
assign SIG0 = tempSIG0;     //set WaveFormer signal to value of tempSIG0
```

Note: You do not have to declare the WaveFormer signal SIG0 because WaveFormer automatically creates it. You DO have to declare the temporary reg **tempSIG0** because WaveFormer doesn't know anything about it.

12.6 Finding and Fixing Simulation Errors

The simulation status button on the button bar displays success or failure information about the last interactive simulation performed. There are two displays, one located on the button bar and the other on the right hand side of the status bar at the bottom of the window. These status displays make it easy to determine when there is a mistake in your Boolean equations or Verilog code. Always make sure that at least one of these displays is visible. The simulation status readout has the following states:

- **Simulation Inactive:** There are no continuously simulated signals defined in the diagram so the simulator is not currently monitoring the diagram. This is the normal state of WaveFormer.
- **Simulation Good:** Simulation was a success and future simulations will be performed if the diagram changes.
- **Simulation Failed:** There is an error in the current diagram. Look in the **waveperl.log** and **verilog.log** files displayed in the Report window for error messages.
- **Simulation Turned Off:** Indicates that the simulation features have been turned off using the **Options > Simulation Preferences** menu. Simulations will not be performed.

There are two different files which are automatically displayed in the Report window that you can view to determine why a simulation failed.

1) Check the **waveperl.log** file for equation syntax errors and unknown signal names in your design equations. If you are using only the Boolean Equation interface, most errors will be easy to find using this file, because it contains a status report from the Perl script that converts the Boolean equation into Verilog code.

2) Check the Simulation Log File **verilog.log** for HDL coding errors. This is the best place to locate HDL coding errors in the user-entered behavioral code or the external models included in the simulation.

The log file also reports the lines in the SynaptiCAD-generated Verilog source code file where the error occurred. The SynaptiCAD-generated source file will have the same filename as your diagram with the suffix TIM appended and an extension of .v instead of .tim (if your diagram is untitled.tim, the source code file is untitledTIM.v). This source file is automatically opened by the *Report* window whenever a file is generated (by default this occurs every time you make a change to your design while simulating signals).

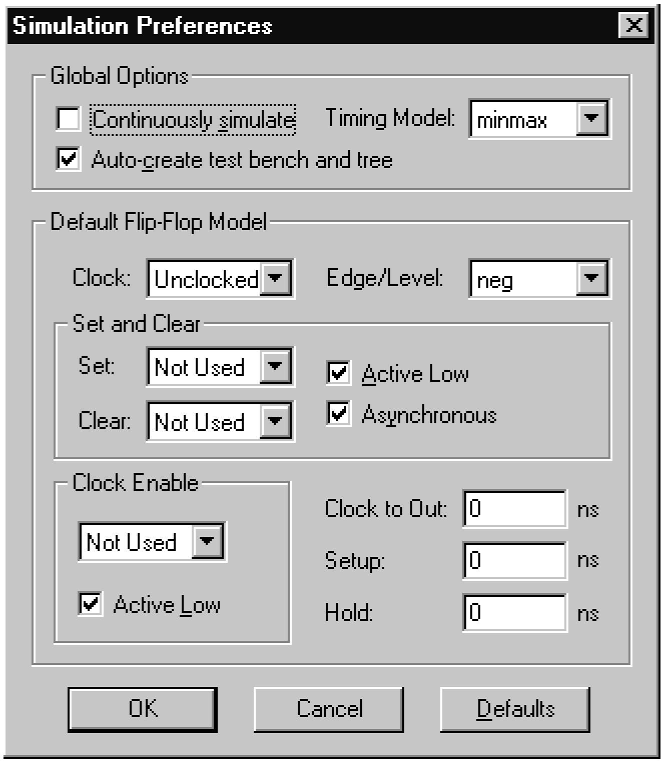
Note: Do not make changes to the SynaptiCAD-generated source file, your changes will be automatically overwritten the next time a simulation is performed. Instead, make the appropriate changes in the *Diagram* window and *Signal Properties* dialog. WaveFormer Pro, VeriLogger Pro, or Test-Bencher Pro will generate and simulate a corrected file.

12.7 Simulation Preferences

The Interactive HDL Simulation performs simulations based on the options set in the *Simulation Preferences* dialog (see Figure 12.1). To open the dialog:

- Select the **Options > Simulation Preferences…** menu option.

The **Global Options** control how simulation is performed in the entire diagram.



Global Options
☐ Continuously simulate Timing Model: minmax
☑ Auto-create test bench and tree

Default Flip-Flop Model
Clock: Unclocked Edge/Level: neg

Set and Clear
Set: Not Used ☑ Active Low
Clear: Not Used ☑ Asynchronous

Clock Enable
Not Used
☑ Active Low

Clock to Out: 0 ns
Setup: 0 ns
Hold: 0 ns

Figure 12.1 Simulation preferences Dialog Box

- **Continuously Simulate** checkbox determines how often simulations are performed. When checked, a simulation is performed each time the design information changes (e.g. when a signal edge is moved). This provides the "Interactive" part of the environment. However, you may wish to turn off this feature during times when you are typing in large blocks of behavioral code into several different signals and you do not want to simulate until you are done with all the changes.

- **Auto-create test bench and tree** check box determines how often the complete simulation model file is created. Each time the design information changes, WaveFormer Pro gathers all the code fragments into a complete simulation model and then performs the simulation. If you wish to edit the *.v file in the report window, uncheck this option to keep WaveFormer Pro from overwriting your changes.

- **Timing Model** drop-down list box determines how delay values are calculated for the entire project. For example, if the user defines a delay D0 with values [10ns, 13ns] and a signal SIG1 which is calculated by a Boolean equation **SIG0 delay D0** then:
  - if **min only** timing, then SIG1 is SIG0 delayed by 10 ns.
  - if **max only** timing, then SIG1 is SIG0 delayed by 13 ns.
  - if **min-max** timing, then SIG1 begins transitioning at 10ns after SIG0 changes and stabilizes at a new value after 13ns. This is the default value.

Note: For Interactive Simulation, both **Continuously Simulate** *and* **Auto-create test bench and tree** must be checked

The **Default Flip-Flop Model** group of options control default timing models for latches and registers used in the project. Each latch and register can be set locally by double clicking on the signal name.

The first two drop-down boxes affect all existing signals and all new signals:

- **Clock:** Names the global clocking signal for the diagram. This is normally **Unclocked**. If you choose a signal as the default clock, all existing and new signals that use the **Clock:Default** setting in the *Signal Properties* dialog will now be clocked by the Global clocking signal. Under this condition you must choose **Clock:Unclocked** in the *Signal Properties* dialog box in order to make a non-registered signal that is described by a Boolean equation.
- **Edge/Level:** Determines whether the default functionality of the Logic Wizard is a register or a latch. For register functionality choose: **neg** for negative edge triggering, **pos** for positive edge triggering, or **both** for edge triggering. For a Latch circuit, choose either **low** or **high** level triggering. The default value is a negative edge triggered register.

The timing parameters only affect new signals. All existing signals will maintain their original timing information.

- **Clock to Out:** Describes the default delay from the triggering of the clock signal to a change on the output edge. This edit box accepts both time values and names of parameters which can contain both min and max timing values. The default value is 0ns.
- **Setup:** Describes the time in which the input must be stable before the clock triggering event. If a min/max time pair is entered, Setup will use the min time. This Setup time is checked during simulation, so any violations will be logged to the **verilog.log** file which is displayed in the Report window.
- **Hold:** Describes the time in which the input must be stable after the clock triggering event. If a min/max time pair is entered, hold will use the min time. This hold time is checked during simulation, so any violations will be logged to the **verilog.log** file which is displayed in the Report window.

The **Set and Clear** section controls the default settings for set and clear lines.

- **Set:** Controls the default signal used for the set line. By default this is **Not Used**. If you choose a default signal for the set line then all new register or latch signals will have a set line.
- **Clear:** Controls the default signal used for the clear line. By default this is **Not Used**. If you choose a default signal for the clear line then all new register or latch signals will have a clear line.
- **Active Low:** If checked, the set and clear lines will control the output when they are low. If unchecked, then the set and clear lines will control the output when they are high.
- **Asynchronous:** If checked, the set and clear lines will control the output anytime that they are active. If unchecked, the model is synchronous and an active set or clear line does not affect the output until the next clock trigger event.

The **Clock Enable** section controls the default signal for the clock enable line

- Clock Enable drop-down list box: By default this is **Not Used**. If you choose a default signal for the clock enable line then all new register or latch signals will have a clock enable line.
- **Active Low:** If checked, the clock will be enabled when the *clock enable* line is low. If unchecked, the clock will be enabled when the *clock enable* line is high.

In the **Delay Mode** area you will be able to specify the type of delays that are being modelled:

- The **Inertial** radio button specifies that a pulse sent that is smaller than the delay will be eaten.
- The **Transport** radio button specifies that all pulses will be transmitted.

The **Flip-Flop Library** drop down list box allows you to specify the parts library that will be used during a simulation.

12.8 Simulation Behind the Scenes

During the simulation process, the Interactive HDL Simulator creates various log and simulation files. You may edit the perl script **tbooleqn.pl** to add functionality to WaveFormer's basic simulation models. You can also use the simulation files with third-party Verilog simulators, including VeriLogger, SynaptiCAD's full featured verilog simulator.

The following is an overview of the simulation process:

1) **HDL Code Fragments:** The user enters equations for each signal into the Logic Wizard section of the *Signal Properties* dialog. Each logic equation is converted to an HDL code fragment by the **tbooleqn.pl** perl script and stored as the **VerilogCode** property of the signal. You can view and edit code fragments by pressing the **HDL code** radio button in the *Signal Properties* dialog box. When the Logic Wizard converts an equation to HDL code, the **waveperl.log** file generates a list of any errors that occur during the translation. This file can be viewed in the report window. Errors are normally caused by improper equation syntax or invalid signal names (signal names like "and" or "76" can be interpreted as Boolean operators or time values).

2) **Model File:** When a simulation is requested, WaveFormer generates a Verilog source file comprised of test vectors for non-equation signals and HDL code fragments for equation-based signals. The Verilog source file has the same name as the current .tim file, but with a **TIM.v** ending.

3) **Model Timing File:** When a simulation is requested, WaveFormer Pro also generates an SDF (Standard Delay Format) file that contains additional timing information (setup and hold times) for the registers and latches in the diagram. The SDF file has the same name as the current .tim file, but with a **TIM.sdf** extension.

4) **Simulation Requests:** Simulations are automatically triggered when design information changes *and* the **Continuously simulated** checkbox in the *Simulation Preferences* dialog is checked (the default setting). Users can also trigger a simulation by pressing the **Simulate Once** button in the *Signal Properties* dialog. Each individual signal has a local **Simulate** radio button in the *Signal Properties* dialog that needs to be checked in order for that signal's code fragment to be included in the simulation model file.

5) **Log & Error File:** The Verilog simulation engine compiles and simulates the model file. During this process, a log file called **verilog.log** is generated. It contains status and error information for the simulation run. If you have modified the HDL code by hand you will want to view this file in the Report Window to make sure that the code simulated without errors.

6) **Register & Latch Models:** The Logic Wizard provides a point-and-click interface for accessing standard register and latch models that are shipped with WaveFormer. The file **wavelib.v** contains the generic register and latch models. Users may view this file, but it is not recommended that they make changes to it.

7) **User Models:** Users can extend the functionality of WaveFormer by adding models to the **waveuser.v** file. Use the **wavelib.v** file as an example of how to add models to the **waveuser.v** file. To view the internal signals of an external model use the full hierarchical Verilog model name and set the **Watch** radio button in the *Signal Properties* dialog.

8) **Simulation End Time:** By default, simulations run to the end of the last drawn signal in the timing diagram. If the inputs to a particular signal are not drawn all the way to the simulation end time, then their last state value is maintained until the end of the simulation. Users can force a specific end time for simulations by dropping a **Marker** and setting it to be of type **End Diagram**.

12.9 Report Window

The Report Window is a general purpose editor that you can use to display and edit ASCII files. During the simulation process the Interactive HDL Simulator creates two error/log files and a simulation file: waveperl.log, verilog.log, and *currentname*TIM.v. These files are automatically displayed in the report window. Switch between the files by pressing the tabs on the bottom of report window.

Normally the report window is used for viewing simulation-related files. However, you can also view and edit other files. However, when a simulation is performed all changed files in the report window are automatically saved at the beginning of the simulation.

The Report Window menu options are:

- **Report > Open Report Tab**: Opens a new file for the report window.
- **Report > Close Report Tab**: Closes the displayed file in the report window.
- **Report > Save Report Tab**: Saves the displayed file in the report window.
- **Report > Save Report Tab As**: Saves the displayed file under a different name.
- **Report > Print Report Tab**: Prints the displayed file.

In addition to the standard editing environment, the Report Window provides the following editing features:

To go to a specific line number, press **<Ctrl>-<Shift>-L**.

To perform a text search, press **<Ctrl>-F**.

12.10 Report Window Editor Commands

The Report Window displays are full featured editor windows. Below is the list of keyboard commands supported by the editor window.

Key	**Function**
arrow keys	Moves cursor one space in the direction of the arrow.
Page Up	Moves one page up.
Page Down	Moves one page down.
Home	Moves cursor to beginning of line.
End	Moves cursor to end of line.
Ctrl+Up	Moves to the beginning of the line or the beginning of the previous line if already at beginning of line
Ctrl+Down	Moves to the beginning of the next line.
Ctrl+Right	Moves to the start of the next word.
Ctrl+Left	Moves to the start of the previous word.
Ctrl+Page Up	Moves to beginning of file.
Ctrl+Page Down	Moves to end of file.
Shift+move combo	Selects between cursor position and position moved to.
F1	Help: show this help file.
F4	Print from window.
Shift+F4	Print Options.
F5	Search
Ctrl+F5	Insert blank line before current line.

F6	Replace
F8	Toggle highlight lines: press F8 and move cursor to end of selection.
F9	Insert blank line after current line.
Shift+F9	Delete current line.
Alt+J	Join line, removes the next carriage return.
Alt+C	Block copy: duplicates selected text inserting it after selection.
Alt+Backspace	Undo
Alt+3	Protected text toggle: selected text cannot be modified while protected.
Alt+7	Change color
Ctrl+T	Tab
Ctrl+A	Select all.
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+F	Search
Alt+S	Find in files (multiple file search/grep)
Ctrl+G	Jump to line#

Chapter 13: Stimulus Generation & Waveform Import

WaveFormer Pro, VeriLogger Pro and TestBencher Pro can export waveform data as a stimulus file for several simulators including VHDL, Verilog, Viewlogic, Mentor, Spice, DesignWorks, and more. These programs can also import data from simulators and PC-based test equipment like the Pod-A-Lyzer and HP's logic analyzers. This chapter is broken up into the following sections.

Stimulus Generation (exporting waveform timing information)

13.1 Export Instructions for Stimulus Generation

13.2 Export VHDL and Verilog Stimulus

13.3 Export List of Simulators and Test Equipment

13.4 Export Script Notes

Importing timing information from simulators and test equipment

13.5 Importing Waveform Instructions

13.6 Import List of Supported Simulators and Test Equipment

13.7 Import from Spreadsheets

The Perl source code that performs the import/export function is shipped with WaveFormer Pro, VeriLogger Pro and TestBencher Pro and can be modified to perform customized actions. SynaptiCAD's ftp site (ftp://www.syncad.com) maintains a repository for scripts written by SynaptiCAD and users.

13.8 Advanced: Modifying Scripts

13.9 Advanced: Adding New Scripts

Background: WaveFormer Pro, VeriLogger Pro and TestBencher Pro use the Perl scripting language to perform most of the importing and exporting of timing information to different formats. Perl scripts can also be used to perform functions on the current timing diagram. Chapter 14 describes how to write your own scripts.

13.1 Export Instructions for Stimulus Generation

To create a stimulus file, use the *Save As* dialog:

- Use the **Export > Export Timing Diagrams As** menu option to open a special version of the *Save As* dialog. This dialog remembers the file type of the last file exported.

- Choose the export format using the **Save as type** list box in the lower left corner of the *Export* dialog box.
- Pick a file name and click the **OK** button. This will produce a file with the timing data in that format.

Note: Once a file is successfully created, it is displayed in the Report window so that you can quickly verify that the file is correct.

To exclude a signal from being exported, use one of the following methods:

A) Double click on the signal name to open the *Signal Properties* dialog. Uncheck the **Export Signal** check box.

B) Select one or more signal names. Select the **Export > Exclude Selected from Export** menu option. This will uncheck the **Export Signal** check boxes in all of the selected signals.

13.2 Export VHDL and Verilog Stimulus

WaveFormer Pro, VeriLogger Pro and TestBencher Pro ship several scripts for exporting stimulus vectors to VHDL and Verilog. All the scripts produce a complete entity-architecture or module test bench that can be directly compiled and linked into an external simulator. The differences between the following scripts are:

- **VHDL transport:** Exports signal transitions as simple VHDL transfer statements.
- **VHDL wait:** Exports signals using wait statements.
- **Verilog:** Exports signal transitions as signal assignment statements.

In order for a signal's waveform data to be exported by the VHDL or Verilog scripts, the signal must be declared as an **export signal** with a direction of **out** (shared output, output, or persistent output). Also, the signal type must be set to match the model which you will be testing. When signals are first added to a project they are defined as export, with a direction of out, and a signal type of std_logic or wire.

To set the export property, direction, and type of a signal:

- Double left click on a signal name to open the *Signal Properties* dialog.
- Check the **Export Signal** check box to indicate that this signal should be included in stimulus generation files.
- Use the **Direction** drop down list box to determine whether the signal is an output or input signal.
    - Any of the **out** directions indicates that the signal is an output of the test bench and an input to the circuit under test. Waveform information will be generated for the output signals.

- Any of the **In** directions indicate that the signal is a response expected from the model under test. Only a port statement will be generated for the input signals (no waveform data).

Note: The stimulus scripts do not differentiate between the different types of **out** and **in** direction types. The direction type is only used by TestBencher Pro when generating interactive test benches.

- Use either the **VHDL** or **Verilog** type drop down list boxes to set the type of the signal. The **VHDL** list box is also an edit box in which you can enter User defined types.

- Use the **Next** or **Prev**(ious) buttons to investigate other signals or click **OK** to close the dialog.

How different information is Exported to VHDL and Verilog:

- **Clock** signals are represented as repeating processes.

- **Signals** are exported with VHDL and Verilog values that most closely match the graphical state. You can change the state-value mapping by editing the VHDL or Verilog perl script.

- **Group Buses** are exported as several signals which represent the member signals. There is no vector that represents the bus.

- **Virtual Buses** are exported as vector signals.

- **Virtual Buses with Virtual State Information:** A signal made up of **valid** segments labeled with virtual state information writes the virtual state information to the export file, instead of the valid state value. This only works with signals drawn with valid segments.

How To Add Virtual State information to a signal: The virtual state information is used to define values for signals whose types do not easily match the graphical state. For example a signal with a type of **integer** or a VHDL user defined enumerated data type of **MYCOLOR** cannot be represented with the seven drawing states available in the program. To add virtual state information:

- Double Click on a valid segment to open the *Edit Bus State* dialog.

- Use the **Virtual** edit box to enter the state of the selected signal.

  - If the state begins with **'b** or **'h**, then it will be interpreted as a language independent binary or hexadecimal value. The size of the bus is determined from the *Signal Properties* dialog Bus MSB and LSB (*Chapter 12: Multi-bit Simulation*).

  - If the state does not begin with **'b** or **'h** then the string is used without any text processing.

  Note: The virtual state will only override the value of a *valid* segment. The other graphical states override the virtual state information

- Use the **Next** or **Prev** buttons to edit other segments on the same signal or click **OK** to close the dialog.

13.3 Export SPICE Stimulus

Signals can now function as virtual Digital-To-Analog Converters for SPICE export. The voltage range of the DAC signal is set by the Logic High and Logic Low Voltage of the signal (set in the analog properties dialog). The output value of the DAC signal at a given time is a function of the Virtual state value of the signal and the high and low voltage levels of the signal.

In the simplest case of a 1-bit signal (MSB=0 and LSB=0), the signal outputs the high voltage when it is at 1 and the low voltage when it is at 0. For an 8-bit signal, the output voltage would be the low voltage when the virtual state is 0,or the high voltage when the virtual state is 255. A virtual state of 127 would output a voltage at the midpoint between the high and low voltage levels.

The rise time and fall time parameters together with the logic switching thresholds in the Analog properties dialog control the slew speed of the DAC as it switches between states.

To change the Analog properties of a signal:

- Double click on the signal name to open the *Signal Properties* dialog.
- Press the **Analog Props** button to open the *Analog Properties* dialog.
- Edit the settings to match your logic family specifications.
- Choose **OK** to close the dialog.

13.4 Export List of Supported Simulators and Test Equipment

WaveFormer Pro, VeriLogger Pro and TestBencher Pro can generate stimulus files for many different simulators. Stimulus file generation is performed using the **Export > Export Signals as** menu item. New simulator scripts are uploaded to SynaptiCAD's ftp site at www.syncad.com. The following scripts are currently available:

Generic Simulation Formats:

- **VHDL transport test bench (*.vhd)** saves the waveforms to a VHDL test bench that uses a single assignment statement for each signal.
- **VHDL wait test bench (*.vhd)** saves the waveforms to a VHDL test bench that uses assignment statements separated by wait statements.
- **Verilog (*.v)** saves the waveforms as Verilog stimulus statements. The resultant file can be used to drive Verilog simulations.
- **SPICE sources (*.cir)** and **(*.cir)s** saves the waveforms in a format that can be read by all SPICE simulators. The entry with the **"s"** is performed using a Perl script that can be edited by the user. The Perl script is slower then the other entry.
- **ABEL stimulus (*.abv)s** saves waveforms to an ABEL test bench.

Standard Simulator Formats:

- **Accolade's PeakVHDL** and **PeakFPGA** packages are VHDL simulators that accept either of the transport or wait state VHDL test benches. Input also supported.
- **ALTERA Vector Format (*.vec)** saves waveforms to a format used by the Altera Max PlusII simulator.
- **Mentor QuickSim II (*.f)** saves the waveforms into a force file that can be read by Mentor's QuickSim II simulator.
- **Minc PLD-Designer(*.stm)** saves waveforms into a format that can be used by Minc's PLD-Designer package.
- **Modelsim Force File (*.sim)** saves waveforms into a force file that can drive signals in Model Tech's Modelsim simulator.
- **OrCAD** simulation is a VHDL simulator and accepts either the transport or wait state VHDL test benches.
- **SPICE digital (*.fst)** saves the waveforms into PSPICE digital format.
- **VHDL Waves Vector file (*.vec)** Waves stimulus standard for VHDL.
- **Workview CMD (*.cmd)** saves the waveforms in a format that can be read by Viewlogic's Workview simulator. Input also supported.
- **Xilinx/Aldec/Orbit (*.asc)** saves the waveforms in the format used by the Xilinx Foundation Design Kit simulator, Aldec, and Orbit's ASIC test vector format.

Test Equipment Formats:

- **HP Pattern Generator (disk) (*hpd)** exports waveforms compatible with HP digital pattern generators including the HP 16522A . Use this entry if you are going to use the "load" command. Detailed instructions are available on-line at www.syncad.com/hpdetail.htm
- **HP Pattern Generator (bus) (*hpd)** exports waveforms compatible with HP digital pattern generators including the HP 16522A . Use this entry if you are going to use the HPIB bus to load file into the Pattern Generator.
- **STIL Test Vectors(*.stl)s** generates test benches in IEEE Standard P1450 Standard Test Interface Language.

13.5 Exporting HP Pattern Generator Stimulus

Since export scripts produce files that are compatible with simulators or test equipment, not all objects or signal transitions result in stimulus events. Many scripts sample signals based on a clock signal. Sometimes Markers or other objects are used to generate special code in the stimulus files. In general, stimulus vectors will not be generated after an End Diagram Marker is encountered.

HP Pattern Generator (disk) (*hpd) and HP Pattern Generator (bus) (*hpd) scripts for the HP 16522A. The most current and detailed instructions can be found on-line at **www.syncad.com/hpdetail.htm**. Below are the basic instructions for drawing a timing diagram that will generate a valid HP Pattern Generator file but may not contain the latest information. The following objects must be included in your timing diagram:

1.A **sampling clock** and user-created signals. Signals are sampled using the first clock in the timing diagram. All signals are sampled at time zero and on each rising edge of the sample clock. Clocks are not exported to the Pattern Generator file.

2.At least **one documentation marker**. The first documentation marker found in the timing diagram generates the *M code line, which denotes the beginning of the main sequence (separating the initialization section from the main). If no documentation marker is present, the *M code is generated automatically and placed at the beginning of the data (just after VECT). Thus, only main sequences will appear (i.e., no initial sequences).

Tips and suggestions for making valid files:

- Single-bit signal states should be drawn with only high and low segments. Virtual buses can contain Valid segments with virtual values that convert to high and low segments. Invalid, Tristate, and empty Valid segments are not supported by the HP16522A and generate an error message during export.
- The state of each signal is converted into a hexadecimal number and written to the file.
- The end of the timing diagram is determined by the last edge of the longest drawn signal or by an **End Diagram Marker** (you must still have at least **one documentation marker** in addition to any End Diagram Marker that you add). If a signal ends early, its last state value is used until the end of the timing diagram.
- There is a limit of 126 signals per stimulus file, and a Group Bus limit of 32 signals per bus. These limits are set by the HP16522A pattern generator.
- The Bus and Disk versions generate basically the same code. The only difference is that the VECT code for the Bus version contains additional information.
- LABEL codes are generated for virtual buses and single bit signals.

13.6 Exporting ALTERA Stimulus

Since export scripts produce files that are compatible with simulators or test equipment, not all objects or signal transitions result in stimulus events. Many scripts sample signals based on a clock signal. Sometimes Markers or other objects are used to generate special code in the stimulus files. In general, stimulus vectors will not be generated after an End Diagram Marker is encountered.

ALTERA Vector Format (*.vec) saves waveforms to a format used by the Altera Max PlusII simulator.

- The timing diagram must have at least one clock and one or more signals with waveforms.
- All signals are sampled at time 0 and on each rising edge of the sample clock. The end of the diagram is determined by the latest time of any drawn signal or an End Diagram Marker. If a signal ends early, its last state value is used until the end of the timing diagram.

- The following table describes how WaveFormer signals are translated into Altera Stimulus files:

WaveFormer Signals	Altera Stimulus
SIG_DRIVING is a single bit signal with direction of out	INPUT SIG_DRIVING;
SIG_RESULT is a single bit signal with direction of input	OUTPUT SIG_RESULT;
ticket[3:0] is a 4 bit signal with direction of out	GROUP CREATE ticket[3..0] = ticket3 ticket2 ticket1 ticket0;

- Draw single bit signals using high, low, and invalid segments. Draw multi-bit signals with valid segments with virtual data that translates into high and low values.

To use an altera vec file:

1) Inside Max PlusII, run the simulator (choose Max+plusII\Simulator menu) to open the *Simulator* dialog.

2) Double click on the file listed as the Simulation Input: *.scf to open the *Inputs Outputs* dialog.

3) Enter the name of the vec file that you generated with WaveFormer Pro into the Input (.scf or .vec) edit box.

4) Click OK to close the *Inputs Outputs* dialog.

5) Click the Start button in the *Simulator* dialog. This performs a simulation and creates an SCF file. The SCF file is the graphical waveform representation used by Altera. From inside Altera's Waveform editor you can add nodes (signals) to view buried and output waveforms.

6) **Note:** The second time you simulate, the Max Plus II simulator is going to use the *.scf file it created from the first simulation. You must explicitly choose the vec file each time you make waveform changes using WaveFormer.

13.7 Exporting STIL Test Vectors

Since export scripts produce files that are compatible with simulators or test equipment, not all objects or signal transitions result in stimulus events. Many scripts sample signals based on a clock signal. Sometimes Markers or other objects are used to generate special code in the stimulus files. In general, stimulus vectors will not be generated after an End Diagram Marker is encountered.

STIL Test Vectors(*.stl) generates test benches in IEEE Standard P1450 Standard Test Interface Language.

- The timing diagram must have at least one clock and one or more signals with waveforms.

- Signals are sampled using the first clock in the timing diagram. Normally the rising edge of the clock is used as the sampling time. However, if the clock's invert property is checked then the falling edge is used as the sampling time. The clock's period property sets the Waveform Table Period in the STIL file.
- In the **SignalGroups** section of the STIL file, a group called **ALL** is defined as:

  ALL = each input signal (black) + each output signal (blue) + each bi-directional inout signal (black and blue)
- A waveform vector **ALL** is generated for each sampling time of the clock.
- Signals are sampled from time 0 to the last simulation time. The last simulation time is determined by the first End Diagram Marker in the diagram. If there are no end diagram markers, then the last simulation time defaults to the time of the last drawn signal event in the diagram.
- If a signal ends early and there is no event at the current sampling time, then the state at that time is recorded as ‘P’ for input (black) signals or ‘X’ for output (blue) signals.
- The names of the **Timing**, **Pattern Burst**, and **Pattern** sections can be changed by editing the first three variables in the stil.epl file.

Note: The definitions of input and output are exactly opposite in the WaveFormer Pro and the STIL language. This is because each format defines waveform direction in terms of what is most important to the format. For example, the STIL language defines waveform direction in terms of the tester (inputs drive the tester, outputs are generated by the tester). WaveFormer defines waveform direction in terms of the generated test bench (inputs come from the device under test, or tester, and outputs are generated by the test bench and drive the tester). The WaveFormer STIL documentation uses input/output as defined by STIL, and also the color that the waveforms appear in WaveFormer input (black) and output (blue). General WaveFormer Pro documentation uses the normal WaveFormer definitions.

- The naming conventions for representing the signal states are:

Input Signals (black) :

'1' => 'ForceUp' (Waverform: high state),
'0' => 'ForceDown' (Waverform: low state),
'N' => 'ForceUnknown' (Waverform: valid or invalid state),
'P' => 'ForcePrior' (Waverform: unknown state),
'Z' => 'ForceOff' (Waveform: tri state),

Output Signals (blue) :

'H' => 'CompareHigh' (Waveform: high state),
'L' => 'CompareLow' (Waverform: low state),
'X' => 'CompareUnknown' (Waverform: valid, invalid or unknown state),
'T' => 'CompareZ' (Waverform: tri state),

InOutput signals (black and blue) : Input and Output signals representation.

13.8 Instructions for Importing Waveforms

To import waveforms from other formats:

- Choose the **Export > Import Timing Diagram From** menu option. This opens a special version of the *Open Timing Diagram* dialog that remembers the type of the last file imported
- Determine the type of script using the **Files of type** list box in the lower left corner of the *Open* dialog. When opening, the file format is automatically detected for some formats using file contents, but it is best to use **Files of type** to specify a particular file format.
- Select a file and click the **Open** button. WaveFormer Pro will use the user-selected script to interpret the file and build and display a timing diagram. The resultant timing diagram can be saved in any format including the *.tim project format.

13.9 Import List of Supported Simulators and Test Equipment

WaveFormer Pro, VeriLogger Pro and TestBencher Pro can import waveform data from several simulators and pieces of test equipment. The importing of waveform data is accomplished in the *File Open* dialog by selecting the correct format using the **Files of Type** list box. The following scripts are supported:

Standard Formats:

- **Verilog Change Dump (*.vcd)** reads waveforms generated by Verilog simulators.
- **Test Vector Spreadsheet (*txt)** reads waveforms generated inside spreadsheets.
- **Xilinx/Aldec Waves Vectors (*.vec)** reads waveforms generated by simulators that support the VHDL Waves format.

Test Equipment:

- **HP Logic Analyzer (*.hpl)** reads data files captured by HP logic analyzers: HP 1660C and1670D benchtop logic analyzers; HP 16500B and 16500C logic analysis system; HP 16550A, 16555D, and 16556D state timing card, in addition to other HP timing modules.
- **Podalyzer Data (*.dat)** accepts waveform files captured by Boulder Creek Engineering's PC-based logic analyzer, the Pod-A-Lyzer.

Simulators:

- **Accolade VHDL (*.awf)** reads waveforms generated by Accolade Design Automation’s PeakVHDL and PeakFPGA simulators.
- **Advanced PLD (*wvf)** reads waveforms generated by Protel Technology's Advanced PLD simulator.
- **DesignWorks (*.tim)** accepts waveform files generated by Capilano Computing's DesignWorks' gate-level simulator.

- **SpeedWave (*.vwf)** reads waveforms generated by Viewlogic's SpeedWave VHDL simulator.
- **SpeedWave (*.vcd)** reads waveforms generated by Viewlogic's SpeedWave VHDL simulator.
- **Workview WFM (*.wfm)** accepts waveform files produced by Viewlogic's Viewsim simulator.
- **Xilinx/Aldec Waves Vectors (*.vec)** accepts VHDL Waves files generated from any compatible simulator including the Foundation series simulators offered by Aldec and Xilinx.

Timing Diagram Formats:

- **Timing Project (*.tim)** signals and parameters from SynaptiCAD products including Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro and TestBencher Pro. To insure file compatibility, open files created with products of the same or earlier version number.
- **TDML (*.tdml, *tdm)** opens timing diagrams and timing parameter tables created in the industry standard on-line data sheet format TDML.
- **Chronology (*.td)** accepts timing diagram files from Chronology's® timing diagram editor.
- **Text Free Parm (*.txt)** libraries saved in the standard spreadsheet compatible file format: comma and tab separated text files that contain a header (first line must be "NAME, MIN, MAX, COMMENT"). This is also compatible with Chronology® library files (formulas are only supported in the td files).
- **Text Free Parm (*.txt)** saves just the free parameters in a spreadsheet compatible format: comma or tab separated with a header (first line must be "NAME, MIN, MAX, COMMENT"). This is the standard way to make a library file. This is also compatible with Chronology® library files (formulas are only supported in the td files).
- **Free Parm (*.fp)** free parameters saved in the Timing Project File Format. Usually a library file.

13.10 Import from Spreadsheets

When your application requires a large number of waveforms that cycle between different modes, it may be easier to generate the waveforms from lists of vector state values rather than drawing each waveform. You can do this by using a commercially available spreadsheet or a text editor to produce waveform files that are compatible with the WaveFormer Pro, VeriLogger Pro and TestBencher Pro Test Vector Spreadsheet format.

WaveFormer Pro, VeriLogger Pro and TestBencher Pro use a very flexible spreadsheet format that describes the waveform's state values. This information is organized into different sections, each section describing a different type of signal, such as a clock or bit vector. A single spreadsheet may contain many sections, and each section may contain one or more signals. The file begins with a free formatted header, followed by any number of sections and comments.

To export from your spreadsheet program:

- Inside the spreadsheet program, save your test vector file as a **tab delimited table**.

Note for Excel users: You must shut down Excel after saving the file as Excel doesn't "share" the file if it is still open.

To import into WaveFormer, VeriLogger, or TestBencher Pro:

- Choose the **File > Open** menu option. This brings up the *Open File* dialog.
- Select **Test Vector Spreadsheet** from the **Files of Type:** drop down list box in the bottom left corner of the *Open File* dialog.
- Select the spreadsheet file and press **OK** to open the file.

Test Vector Spreadsheet Format

The header may contain any type of information. WaveFormer Pro will ignore any text until it encounters the first section. A section ends when a new section begins. Sections begin with a section header in the first column. All section headers are single words, first letter capitalized, and wrapped in square brackets (e.g. [Comment]). Any parameters for that section are on the same line as the header, one per cell, as defined by individual section types. The lines following the section header contain information about the section (blank lines are ignored). Any number of sections can be used, in any order. An undefined section header is treated as a comment section. Sections and signal names are case sensitive.

Section Types

[Clocks]

Declares clock signals. The first line after the section header should have the clock attribute titles: **Name**, **Period**, **Offset**, **Duty**, and **Invert**, in that order. The remaining lines contain the names of clocks and their attribute values, each in a column. Period, Offset, and Duty are numeric values. Invert is a Boolean value (1 for true, 0 for false).

[Comment]

Declares a comment block. Anything inside this section is ignored.

[End]

Ends input. When read by WaveFormer Pro, VeriLogger or TestBencher Pro, this section header is equivalent to the end of the file. No section beyond this point will be read.

[Vectors]

Parameters: End=# Extends signals out to this time. Defaults to 50 past the last event if not used.

Description: A list of timed events for a list of signals.

The line following the section header must be a title line. The title line consists of **Absolute**, **Relative**, a listing of the signals used in this table, and **Comment** (each item in its own cell). Note that a signal cannot be in two different vector sections, this will cause two signals with the same name to appear in WaveFormer.

Signals are defined in the form **[@ / &]signalname[[lsb:msb]]**. Each signal name starts with an optional special character that determines it's direction in when outputted to certain formats. The '**&**' in front causes the signal to be an input, '**@**' causes the signal to be an output. If no character is used, it is considered an input. At the end of the signal name is the size of the vector. It is given in [lsb:msb] form (i.e. [0:7] is an eight bit vector). If no size is given, it is assumed to be a single bit vector. Neither the direction or the size is considered part of the name.

Each additional line is a list of all of the signal states at given times. The first column, Absolute, gives the time from the beginning of the simulation. Times must be in ascending order. Relative is most often used as the amount of time since the last event (from which absolute is calculated using an Excel formula), but is ignored by WaveFormer. Each signal has a column where its state at that time is stored. Valid states are 1, 0, X, Z, H, and L, in upper or lower case. After the last state is an optional comment. Anything after the last signal is ignored, so multi-cell comments are allowed. Any line that does not start with a time is considered a comment, so comments between times are allowed if there is nothing in the first cell. The signals will be drawn to 50 units beyond the last time given, or to the time given by the End parameter.

Example SpreadSheet:

This is the Header section of the spreadsheet. You can put any title you like here. In the next section the END=50 parameter will cause the signals to be extended out 50 display time units beyond the last event time. SIG1 has a bus width of 8 bits.

[Clocks]						
Name	Period	Offset	Duty	Invert		
CLK1	20	5	10	0		
CLK2	25	10	15	1		
[comment]						
This	section	is ignored.				
[Vectors]	End = 50					
Absolute	Relative	&SIG1[0:7]	@SIG2	&SIG3	@SIG4	Comment
0	0	1	0	x	1	Start simulation
5	5	2	x	1	0	

10	5	4	z	1	0	Some data
15	5	8	z	0	1	
20	5	16	z	0	1	
25	5	32	z	0	0	End it
[End]						

13.11 Advanced: Modifying Scripts

Waveperl scripts are normal text files that can be edited using your favorite text editor or Notepad. The standard scripts that ship with SynaptiCAD products contain comments that identify the most probable locations for user modifications. Before you make changes to a script we recommend that you make a backup of the original file. We also recommend that you read *Chapter 14: Writing Waveperl Scripts*.

13.12 Advanced: Adding New Scripts

When you design your own script, you will have to tell WaveFormer Pro that the script exists so that your script will appear in either the **Save File as Type** list box of the *Export* dialog (export script), or the **List of Files Type** list box of the *Open* dialog (import script), or the **Export** menu (dynamic script). To add a script to WaveFormer Pro:

- Select the **Export > Add/Execute Script...** menu option. This will open a dialog of the same name.

- Select the type of script to be added: Dynamic, Export, Import, Conditional, and Action.

- Type the script name in the **Script Name & Command Line** edit box. By convention, export script names end with ".epl", import script names end with ".ipl", and dynamic, conditional, and action script names end with the extension ".pl".

- For import and export scripts, type a file filter into the **Filter** edit box. The filter is used by the common dialog boxes to display files of that form. A filter of *.tim will display all the tim files in a specific directory. Dynamic scripts do not have filters since they do not work directly with files.

- Type the description of the script into the **Menu Description** edit box. It is best to keep the description under 19 characters long because Microsoft's common dialogs chop off the rest. An example of a description is **Pod-A-Lyzer Data (*.dat)**.

- Press the **ADD** button to add the new script to the list.

- Choose the **OK** button to close the dialog.

Chapter 14: Writing Waveperl Scripts

Waveperl is an extended version of the Perl language that contains additional functions for manipulating data structures inside WaveFormer Pro, VeriLogger Pro and TestBencher Pro. Waveperl scripts are compiled and executed by a perl interpreter embedded into WaveFormer (actually it is not, strictly speaking, an interpreter since your script is really compiled on-the-fly). The functions added to Waveperl that support manipulation of objects in WaveFormer are collectively referred to as the WaveFormer API.

In order to write a Waveperl script you will have to learn the basic syntax of the Perl language and become familiar with the functions available in the WaveFormer API. The rest of this chapter is devoted to introducing Perl and hints on writing and debugging Waveperl scripts. After reading the chapter, study one of the standard scripts that is shipped with WaveFormer. Two good example scripts to study are **spice.epl** and **tvhdl.epl**. You may also find it useful to print the WaveFormer API documentation (**twfapi.txt**) and the on-line Perl documentation. Finally to learn general Perl syntax and practical programming tips, we highly recommend the book *Programming Perl*, 2nd Edition by Larry Wall (Perl's creator), Tom Christiansen, and Randall Schwarz.

14.1 Why a scripting language?

1) The built-in scripting language allows WaveFormer Pro to potentially import or export to any waveform format. If you use a format that is not currently supported, you can write your own scripts.

2) Scripts make it easier for SynaptiCAD to upgrade WaveFormer Pro capabilities and distribute the upgrades to you, the end-user.

3) End-users can change scripts as desired to meet their own particular needs (for example: You're designing an IC that uses a 3.3V power supply and you want your SPICE test vectors to represent a high state as 3V instead of 5V).

14.2 What is Perl?

Perl is a popular scripting language for text processing. Perl originated in the UNIX environment and has since been ported to DOS & Windows environments. Perl's power and flexibility have made it the most widely used language for Web-based programming. Perl has several advantages over regular programming languages:

1) Perl compilation is extremely fast for reasonably sized scripts, making it a good fit for the kind of iterative programming required when writing an import or export script.

2) Perl was originally created for text processing and report generation, so it simplifies many tasks associated with file and text processing. Most waveform files are stored in an ASCII (text) format. Perl also has more than adequate support for binary file formats. Perl's text processing capabilities also make it an excellent language for writing code generators such as TestBencher Pro.

3) Perl is object-oriented. This makes it possible to do a direct mapping from WaveFormer's internal functions (written in C++) to Perl functions. It also means you can write your own objects in Perl.

4) Perl is extensible. Many Perl modules that extend the capabilities of Perl have already been placed in the public domain, and more are being written all the time. See the Perl documentation for information on available Perl modules.

5) Perl is fun! Although Perl syntax may first appear cryptic due to the use of $, @, and % in variable names, you will quickly come to appreciate the ease with which you can write code that would require hours or even days in C or C++.

14.3 Perl data types

A Perl variable can be one of three data types: scalar (contains a single value), array (array of scalar values indexed using an integer), and associative arrays (array of scalar values indexed using a string). In Perl, the type of a variable dictates the first character of its name.

- Scalars begin with a **$** (e.g. $myValue)
- Arrays begin with a **@** (e.g. @myArray)
- Associative arrays (sometimes referred to as hashes) begin with a **%** (e.g. %myHash).

Scalar values can store character strings, numbers, and references (the Perl equivalent of pointers). Scalar values are automatically converted between strings and numbers depending on the function they are passed to. Arrays contain an indexed list of scalar values. An associative array contains a set of key-value pairs. A scalar value in an associative array is accessed by specifying the associated string key.

14.4 Perl's global variables

Perl contains a number of predefined global variables. To distinguish these variables from user-created variables, the predefined variable names are all two characters where the first character sets the data type ($,@,or %) and the second character is a non alphabetic character.

The most important predefined variable is **$_**. $_ is the default scalar upon which many functions automatically operate if the functions are called without parameters.

For example, the statement:

```
$l = <STDIN>;
```

reads a line of characters from standard input into the variable $l. If the result of <STDIN> wasn't assigned to $l:

```
<STDIN>;
```

then **$_** would be assigned the line of characters.

The second most important predefined variable is **@_**. @_ is the array used to pass arguments to subroutines. At the beginning of most subroutines you will see a statement similar to the following:

```
    my ($var1,$var2) = @_;
```

This copies the first argument from the subroutine call into $var1 and the second argument into $var2.

14.5 Using variable interpolation in Perl text strings

Text string literals in Perl can be surrounded by either single quotes or double quotes. Text strings surrounded by single quotes are taken literally without any translation (except for \' and \\ in order to allow single quotes and back slashes to be placed in a single quoted string). Text strings surrounded by double quotes, however, are subject to backslash and variable interpolation.

Backslash interpolation means that character sequences such as \n (newline) and \t (tab) are translated by Perl to the appropriate ASCII code. Variable interpolation means that Perl variables embedded in the string are automatically replaced with their values. For example, if a variable named $state had a value of **’h1111**, the string literal:

"ADDRESS <= $state after 10 ns"

would be translated by Perl to:

'ADDRESS <= ’h1111 after 10 ns'

14.6 Pattern matching (regular expressions)

The most common way to search and change text (strings) in Perl is using regular expressions. Regular expressions are constructs used to define a pattern to match a string against. They are commonly used in many Unix programs. If you are not familiar with regular expressions, you should read the perl documentation on regular expressions. Since perl adds some enhancements to normal regular expressions you should probably look at this section even if you are already familiar with regular expressions.

Tip on writing regular expressions: it is often easier to code a complicated regular expression than it is to read and interpret it later, so it's a good idea to document the purpose of a regular expression unless it is extremely simple.

14.7 Notes on writing import/export scripts

When running an import script (executed from **File > Open** menu option), the file specified by the user in the *Open File* dialog is redirected to standard input. This means that any data read from standard input by your import script is actually read from the file specified by the user.

When running an export script (execute from **File > Save As** or **Export > Export As** menu options), standard output is redirected to the file specified by the user in the *Save As* dialog. Whenever you print something in your export script without specifying a file handle it is printed to standard output (the export file specified by the user).

14.8 Tips on debugging Waveperl scripts

Error messages during compilation or execution of a Waveperl script are redirected to a file called **waveperl.log**. When debugging a new script it is a good idea to constantly watch this file! Generally you will want to have at least three windows up when debugging your script: one for WaveFormer Pro, one for your script, and one for **waveperl.log**.

When debugging an export script, the simplest way to view debugging information is to temporarily add extra print statements to your script that will show up in the exported file.

When debugging an import script, the simplest way to view debugging information is to print debug info to **waveperl.log**. To do this, add the statement:

```
select STDERR;
```

to the beginning of your Perl import script. This will redirect the output of print statements to waveperl.log (don't do this for an export script as it will redirect your export output). Of course, you can also open your own file using Perl and print error messages to it using its file handle.

To make a change to your Perl script and execute the new script, modify the script file, save your change, and re-execute the script from WaveFormer. By default, you will have to shut down WaveFormer before it will recognize your changes because WaveFormer caches the scripts it executes. Therefore, before you begin debugging your script you should probably turn off script caching. With script caching off you do NOT have to restart WaveFormer each time you change your script, because WaveFormer will dynamically recompile it.

To turn off script caching, select the **Export > Edit Object Properties** menu option, and add an Application Property named **DoNotCacheScripts** with a value of **True**. Be sure to delete or set this property to false after you have finished debugging your script as this option significantly slows down initial script execution speeds (they are recompiled each time they are run).

14.9 Perl and WaveFormer Pro API Documentation

Perl: Before you write your first script we recommend you read the first section of the on-line Perl documentation included with WaveFormer (perldoc.zip). This documentation is in **html** format so you'll need a web browser such as Netscape or Mosaic to view it (the documentation is also available in other forms at the Perl CPAN site). You'll also probably want to print the Perl quick reference manual (**perlref.ps**) and the WaveFormer API documentation (**twfapi.txt** lists functions internal to WaveFormer Pro that can be called from a Perl script). Next, study the scripts shipped with WaveFormer Pro. The easiest way to begin a new script is to use one of the pre-written scripts as a template.

There are several books on Perl that are available in most bookstores. We also maintain links to several popular sites for information on Perl at our Web site (**http://www.syncad.com/**).

WaveFormer Pro API: The file **twfapi.txt** contains a description of functions in WaveFormer Pro that can be called from Waveperl scripts. This file is only shipped with the full version of WaveFormer Pro.

Sharing your scripts with others: If you do write a script and you think it may be useful to other users, we encourage you to upload it to the incoming directory of our ftp site (www.syncad.com). If you do, send an email to sales@syncad.com to let us know about it so that we can place it in the appropriate directory. It is also important to place a statement in your script that you are placing it in the public domain so that other users can freely use it.

14.10 Object Properties

Object Properties are properties that can be attached to objects in the timing diagram and accessed by Waveperl scripts. An object property consists of two text strings, a name (spaces not allowed) and a value (spaces allowed). Object properties allow the user to represent and store information needed by export scripts that WaveFormer Pro was not programmed to handle (e.g. data type information for signals being exported to a VHDL test bench).

To add a property to an object in the timing diagram:

- Choose the **Export > Edit Object Properties** menu option. This opens the *Object Properties* dialog box.
- Choose the type of object that you want to work with from the **Object Type** drop-down list box at the top of the dialog. This causes the names of all objects of that type to be displayed in the Object Name list box.
- Select the object to which the property is to be added.
- Type the name of the new property into the Property edit box and its value into the value edit box. An example property for a signal object might have the name “VhdlType” and a value of “integer”.
- Click the **Add** button to add it to the signal.
- Click the **OK** button to close the dialog.

The VHDL wait script (wvhdl.epl) demonstrates how property-value pairs can be used to extend the capabilities of WaveFormer Pro. The script uses the signal properties, VHDLType, to define the type (integer, bit_vector, etc.) of stimulus signals. The user can set this property directly in the **Export\Edit Object Properties** dialog (it is also set indirectly when a VHDL type is selected in the *Edit Signal* dialog**)**. If the VhdlType property is defined for a signal then the script assumes a type of **std_logic**.

NOTE: Object properties associated with a timing diagram object are supplemental to the normal state of the object. WaveFormer Pro does not attempt to interpret object property information. It only saves the information and provides an interface for adding and changing the information. Waveperl scripts have access to object property information, and it is the script writer's job to define the meaning of object property information within a particular script.

Future versions of WaveFormer Pro will likely make use of some property names. These property names will begin with a '$' to distinguish properties that are interpreted by WaveFormer Pro. For this reason, we recommend that you do NOT use property names that begin with a '$' (e.g.$Internal).

14.11 Script Naming Conventions

There are several types of scripts which perform different types of actions. By convention the name of the script indicates the type of script.

- **Export Script** names end with ".epl". They take objects that exist inside the current timing diagram and perform a mapping to a new file format (e.g. converting signals into stimulus files for simulators).
- **Import Script** end with ".ipl". They read an external file and convert that information into objects in the current timing diagram (reading a waveform file created by the Pod-A-Lyzer and converting it into waveforms inside WaveFormer Pro).
- **Dynamic Script** names end with the extension ".pl". They perform functions on the current timing diagram. Both the Temporal and Boolean equations were originally developed using dynamic scripts. They provide a way for the user to add functionality to WaveFormer Pro.
- **Conditional** and **Action** scripts are all contained inside verilog.pm and vhdl.pm files. The names inside the *Add Script* dialog indicate the routine name instead of the file names that are used by the other scripts. These scripts describe different conditions and actions that a Sample parameter can trigger on or act on inside TestBencher Pro.

Chapter 15: Analog Import and Export

The analog features allow for the import, export, and manipulation of analog signals. Digital signals can be converted to analog signals and vice versa (similar to the way analog-to-digital converters function).

15.1 Displaying Analog Signals as Analog Waveforms

’Analog Display’ capabilities can be used with signals and virtual buses in several ways. Signals with binary, decimal, hexadecimal, and real radixes can use the analog display. If the signal is not using the real radix, the analog display will take those values, determine the possible range of numbers based on the bus size and display each segment as a line at a relative height to that range.

For example, if you have a 4-bit bus, the range of possible values for that bus would be from 0-15 in decimal. So the analog display would show a zero as a line at the bottom of the signal and a 15 would be shown at the top. If the radix is real, the range is determined by the ’Logic High’ and ’Logic Low’ voltages in the *Analog Properties* dialog box. Values outside of the current high and low range display as an empty box on the waveform.

The size ratio can be used to increase the vertical size of signals. Using the size ratio, a signal’s height can be specified as an integer multiple of the default signal size. This feature is often useful when displaying analog signals.

15.2 The Analog Signal Property Dialog

The *Analog Properties* dialog allows you to set properties that will replicate the specifications of the equipment you are using.

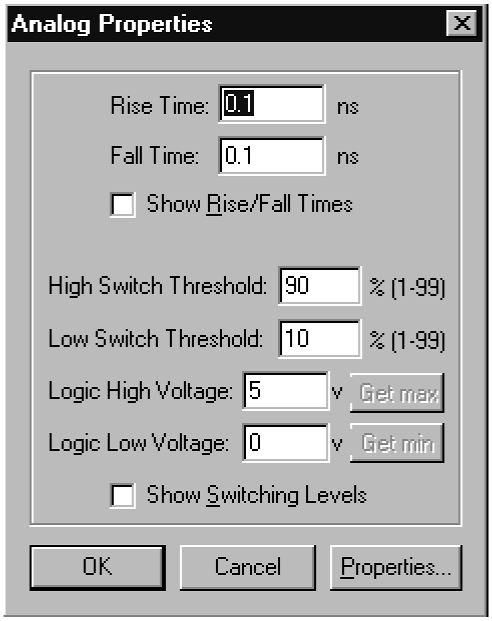
To Open the *Analog Properties* dialog:

- Double click the signal you wish to modify. This will open the *Signal Properties* dialog.
- Make sure the **Analog Display** checkbox is checked.
- Click the **Analog Props** button. This will open the *Analog Properties* dialog.

The *Analog Properties* Dialog

- The **Logic High Voltage** and **Logic Low Voltage** edit boxes allow you to specify the maximum values of the analog signal.
- The **High Switch Threshold** and **Low Switch Threshold** allow you to determine a high and low ’detection limit.’ The percentages entered in these edit boxes determine the point at which the value will be read as the voltage high or voltage low value.

- The **Rise Time** and **Fall Time** edit boxes can be used to specify the amount of time it takes for the signal to go from the High Switch Threshold to the Low Switch Threshold.



Rise Time: 0.1 ns
Fall Time: 0.1 ns
☐ Show Rise/Fall Times
High Switch Threshold: 90 % (1-99)
Low Switch Threshold: 10 % (1-99)
Logic High Voltage: 5 v
Logic Low Voltage: 0 v
☐ Show Switching Levels

Adjusting the Height of Analog WaveForms

- Open the *Signal Properties* dialog by double clicking the name of the signal to be modified.
- Enter the desired value in the **Size Ratio** edit box.
- Click the **Apply** button to view the change.

Note that this does not change any of the values of the waveform, only the manner in which it is displayed.

15.3 Quantizing Analog Signals (Real Radix) Into Digital Signals

The conversion of an analog signal to a digital signal can be performed using the *Signal Properties* dialog. The upper and lower bounds of the analog signal are determined by the ’Logic High’ and ’Logic Low’ voltage settings in the *Analog Properties* dialog box. The precision of the conversion will be determined by the bus size. For instance, if the bounds (’Logic High’ and ’Logic Low voltage settings) are set to 0 and 15 and you are converting to a binary radix, selecting a bus size of 4 would allow you to represent the integers 0 to 15. The greater the bit size, the greater the precision of the converted signal. Note: this "quantization" of floating point values into digital values is the same process performed by an Analog To Digital Converter (A/D Converter), so it can also be useful in modeling systems containing A/D Converters. Values less that the lower bound are "clipped" to 0, and values greater than the upper bound are "clipped" to the largest binary value that can be represented by the given bus size.

Conversion from a digital to analog can also be performed. The default settings for ’Logic High’ and ’Logic Low’ are 5 and 0, respectively. You can set these values manually (in the *Analog Properties* dialog box). or have the values determined from the signal data. The **Get min** button can be used to automatically search the state values in the waveform and set the ’Logic Low’ value to the lowest value found. The **Get max** button will perform the same function for the ’Logic High’ setting, finding the highest value on the signal waveform.

15.4 Exporting Analog Signals as SPICE Stimulus

Signals can also be exported as SPICE voltage sources to provide stimulus for a SPICE netlist. Signals with hexadecimal and binary radixes are converted to decimal numbers before being exported.

Signals with ’real’ radixes are exported without conversion. These signals can model simple digital stimulus for digital inputs to a SPICE model or they can serve as virtual arbitrary waveform generators for driving analog inputs of a model. The ability to generate digital stimulus is particularly handy when using SPICE to analyze the timing and switching characteristics of a digital logic block.

Signals that are exported as analog stimulus will be converted to the selected format automatically.

To Export Analog Stimulus:

- Select the **Export > Timing Diagram As...** menu option. This will open the *Save As* dialog.
- Select the file type to export from the **Save as:** drop down list box.
- Type the name of the file to export in the **Filename:** edit box.
- Click the **Save** button to export the timing diagram.

15.5 Importing SPICE Transient Simulation Waveforms

The time-based waveform results of SPICE transient simulation runs (*.csd) can be imported into Waveformer. When signals are imported from a SPICE simulation, all of the values are read in as ’real’ values. The **Analog Display** checkbox can be checked to view the waveform as an analog signal. You can also quantize the signal as described in Section 15.3 and use the **Analog Display** checkbox. Note that the logic high and low voltages in the *Analog Properties* dialog box are calculated automatically from the waveforms loaded from **.csd** files (these initial voltage settings can be overridden if desired), so this step can be skipped.

To Import SPICE Waveforms:

- Select the **Export > Import Timing Diagram From...** menu option. This will open the *Open* dialog box.
- Select the **SPICE - CSDF Format (*.csd)** option from the **Files of Type:** drop down list box.
- Select the file you wish to import.
- Click the **Open** button to import the file.

Chapter 16: Waveform Comparisons

Waveform comparisons graphically display the differences between compared waveforms for two timing diagrams or individual signals. This feature is very useful when comparing two different simulation runs, as well as for comparing logic analyzer data to a simulation run. The specific regions where waveforms differ turn red when the two waveforms are compared. A range of **tolerance** can be provided using the **compare** signal settings. A time-ordered list of the differences is also reported in the difference tab of the report window (VeriLogger and TestBencher only).

16.1 Compare Signals

Compare signals are specified using the *Signal Properties* dialog box. These signals are used to find the differences between two waveforms. Initially, these signals are designated with a blue signal name and a black waveform. When a waveform comparison is performed, the signal label for the compare signal turns red if differences have been found. The specific differences that have been found during the comparison turn red on the compare signal.

When comparing two waveforms, the signal names must match. This can be accomplished by changing the names of the **compare** signals in the *Signal Properties* dialog (see below) or by ensuring that the signal names in the two files match. If a compare signal is specified that does not have a companion signal (a signal with a name specified), then the signal will not have anything to be compared with.

16.2 Comparing Individual Signals

A signal can be changed to a **compare** signal to be used for comparison. This method works for both unmatched signals in files that you are comparing and for signals that you have created in this file. Any difference between signals that are being compared appears in red on the timing diagram. Multiple pairs of waveforms can be compared in this manner. To compare two signals, they must have the same name and one must be of type **compare**.

- Double click on the signal name to open the *Signal Properties* dialog.
- Change the name of the signal to match the signal you want to compare with. This will not change the signal name in any other file, only in the file that you are working with.
- Select the **Compare** radio button located in the top part of the dialog. This makes the *Signal Properties* dialog display the tolerance controls used to define how the compare will be done.
- Next, push the **Compare** button to run the comparison.

16.3 Comparing Timing Diagrams

The timings diagrams that are being compared can be the result of two different simulation runs, or one or both could contain data acquired by a logic analyzer. To compare two timing diagrams:

- Load in the first timing diagram. Either use the **File > Open Timing Diagram...** menu option to load a new file or use the current timing diagram.
- Choose the **File > Compare Timing Diagram...** menu option to load in the signals to be compared. This opens a *File* dialog that lets you select the file to be compared. Closing this dialog loads the second set of signals and sets their signal type to compare. Any two signals that have matching names will automatically be compared. The compare signal will appear under the original signal.

16.4 Display Options for Compare Signals

By default, the two sets of signals being compared will be interleaved. The original signals will be displayed with the comparison signals immediately following. If two diagrams are being compared, this pattern will follow for every individual signal, so that the signals are interleaved.

You can have the original signals and the compare signals grouped into separate sets instead. To group the signals in this manner:

- Make sure the **View > Compare >Interleave Compare Signals** menu option is not checked.
- Select the **File > Compare Timing Diagram** menu option.

16.5 Adjusting Comparison Tolerances

Ranges may also be used for comparisons. Instead of looking for comparisons directly on an edge, you can allow a tolerance within a set range of the edge. To set the tolerance of a **compare** signal:

- Open the *Signal Properties* dialog by double clicking the signal name. Note: the tolerance can *only* be set on the **compare** signal - not on the original signal.
- Specify the tolerance range previous to the edge by typing a value into the **-Tol:** text box. (This value will be in ns.)
- Specify the tolerance range after the edge by typing a value into the **+Tol:** text box. (This value will also be in ns.)
- Click on the **OK** button to close the dialog. Now when you run the comparison a tolerance will be provided as specified.

The tolerance for multiple signals can be set simultaneously:

- If the *Signal Properties* dialog is open, close it.

- Left click the signal names in the signal window to select the signals to be edited. Notice that each signal can be selected individually.
- Right click one of the highlighted signal names and select **Edit Selected Signal(s)** from the pop up menu.
- Proceed with the steps outlined above for editing signals.

16.6 Modifying All Comparison Signals Simultaneously

It is often handy to be able to modify the properties of all compare signals within a timing diagram when working with the comparison of multiple signals. This allows you to set a standard tolerance for the comparison, for example.

To Edit the Properties of all Compare Signals in a Diagram:

- Select the **View > Compare > Edit Compare Signals** menu option. This will open the *Signal Properties* dialog with all compare signals selected for editing.
- Edit the property settings that you want to be uniform for all compare signals.
- Click the **OK** button to apply the settings and close the *Signal Properties* dialog.

Appendix A: Menu Listings

File Menu

New Timing Diagram: Clears the current timing diagram. Equivalent to restarting the program.
Open Timing Diagram: Opens an existing timing diagram.
Merge Timing Diagram: Opens a second file and adds its contents to the current timing diagram. Duplicate signal and normal parameter names are changed. Duplicate free parameters overwrite existing free parameters, so always make corrections to the **free parameter library** files before loading into a project.
Compare Timing Diagram: opens a timing diagram and compares signals with the same name (waveform comparison). See Section 6.4: Waveform Comparisons.
Save Timing Diagram: Saves the current timing diagram.
Save Timing Diagram As: Saves the current timing diagram with a new file name.
Print Diagram: Prints the Diagram window (covered in Chapter 9).
Print Parameters: Prints the Parameter window.
Printer Setup: Changes default printer setting.
Exit: Closes the program.

Export Menu

Import Timing Diagram From: Imports waveform data from a variety of sources as a timing diagram.
Export Timing Diagram As: Used by WaveFormer, VeriLogger and TestBencher Pro to export all signals that are not marked for non-export. This menu option opens the export dialog. Select the desired export format from the Save File as Type control (located in the lower right-hand corner of the dialog).
Export Timing Diagrams in Project: WaveFormer, VeriLogger and TestBencher Pro use this to export all timing diagrams associated with a project.
Generate Test Bench: TestBencher Pro uses this dialog to generate the HDL code for the top-level test bench of a project (See the TestBencher Pro manual).
Edit Object Properties: Opens a dialog that lets the user attach **object properties** to the current timing diagram. This is an advanced feature that is used by designers who are writing their own scripts for WaveFormer Pro, VeriLogger Pro, or TestBencher Pro (Chapter 14).
Exclude Selected from Export: All the signals that are currently selected will be excluded from export operations (marks signals as non-exported). This menu item is only active when the name of at least one signal is highlighted.
Show Non-Exported Signals: Displays the names of signals that have been marked for non-export and allows the user to optionally mark the signals for exporting. This menu item is only active when at least one signal has been marked as non-exported.
Add/Execute Script: Opens a dialog of the same name and allows you to add new perl scripts which will display themselves in the *Open*, *Save As*, and *Export As* dialogs. It also allows you to dynamically execute perl scripts (*Chapter 13: Adding New Scripts* and *Chapter 14:Writing Waveperl Scripts*).
Search and Replace Signal Names: Opens a dialog that allows you to replace signals names by matching a name expression.
Export from Waveform into VHDL:
Export from waveform into Verilog:

Edit Menu

Undo *action*: Undo last action performed.
Redo *action*: Will redo the last undo *action*.

Delete: Deletes the selected object.
Undo Delete: Undoes the last delete.
Clear Red Events: Deletes all same state transitions, which are displayed as red rectangles or spikes. Same state transitions occur when a signal segment is changed to the same state as a neighboring segment. These can also be individually deleted by selecting the transition and pressing the delete key.
Copy OLE Image To Clipboard w/ save: Copies the timing diagram to the clipboard as an OLE object for use with another OLE enhanced program (available with OLE Option for Windows).
Copy OLE View To Clipboard w/ save: This opens a dialog that allows you to specify the specific View that you would like to store. See Section 9.6 for more information (available with OLE Option for Windows).
Copy To Clipboard: Copies the current timing diagram to the clipboard as a bitmap.
Cut Signals/Text: Removes the selected signals or text a places it on the clipboard
Copy Signals: Copies the selected signals and any attached parameters or text to the clipboard.
Paste Signals: Pastes any signals in the clipboard into the drawing.
Block Copy Waveforms: Copies a section of waveforms and pastes the sections either onto (overwrite) or into (insert) any signal in the diagram (*Chapter 1:Block Copy Waveforms*).
Copy To Clipboard: captures the current window. See **Clipboard** in Chapter 9.
Edit Clock: opens the *Clock Properties* dialog. This menu item is active when a clock name is selected.
Insert Clock Cycles: Insert clock cycles to the right of the selected clock edge.
Delete Clock Cycles: Delete clock cycles to the left of the selected clock edge.
Right Click Delete Mode: When checked you can delete objects with 2 mouse clicks: 1) left click to select the object and 2) right click to delete the object.
Edit Text: Edit the selected text object's attachment, font, or text string.
(Un)Lock Edges of Selected Signals: Either locks or unlocks all the edges of the selected signals.
Edit Waveform Edges: Allows you to modify all edges on a selected set of signals according to an equation you provide.

Bus Menu

Add Bus: If signals are selected, this command creates a group bus from the selected signals. If no signals are selected a new group bus is created.
Expand and Delete Bus: Deletes a bus and displays the member signals.
Group Bus <-> Virtual Bus: Transforms a group bus to a virtual bus.
Align to Bus Edge: Move all nearby member signal transitions to the exact time of the selected bus edge.
Bind Group Bus Edge: Add invisible links to all the member signal transitions that occur at the same time as the selected bus edge. This means that if a member signal transition is moved all the other members will also move.
Unbind Group Bus Edge: Remove the invisible links between the member signal transitions at the same time as the selected bus edge.
Create User-Defined Radix: Opens a dialog that allows you to create a new radix. This allows you to specify meaningful names for state values.

Libraries Menu

Parameter Library Preferences: Opens the *Parameter Library Preferences* dialog which contains the Library search list.
View Parameter Library Parts: Opens the *View Parameters in Libraries* dialog and allows the user to view the free parameters contained in the libraries on the library search list.
Copy Referenced Library Parameters into Table: Copies the library free parameters that are currently being referenced into the project.
Update Project from Parameter Libraries: Replaces project timing values with the library values, for each parameter in the project has the same name as a library free parameter.

Save Free Parameters as Library: Saves only the free parameters to a library file.

Save All Parameters as Library: Saves all the parameters and free parameters to a library file (no waveforms or signals are saved).

Macro Substitution List: Opens the *Edit Formula Macros* dialog.

View Menu

Zoom In: zooms in to show more detail.

Zoom Out: zooms out to show less detail.

Show Delays: If checked, delays are shown in the Diagram window. Default is checked.

Show Holds: If checked, holds are shown in the Diagram window. Default is checked.

Show Setups: If checked, setups are shown in the Diagram window. Default is checked.

Show Samples: If checked, samples are shown in the Diagram window. Default is checked.

Show Text: If checked, text objects are shown in the Diagram window. Default is checked.

Show Hidden Text: Allows attachments to signals like text objects and gridlines to continue to be displayed even when their parent signal is hidden (*Chapter 8: Diagram Window Display*)

Show Critical Paths: If checked, delay objects are color coded to indicate which edges of the delayed transition are set by the delay. Default is checked (*Chapter 5: Add Delays*).

Show Grid Lines: If checked, grid lines on signals and clocks will be shown in the Diagram window.

Hide Selected Signals: Hides selected signals. If no signals are selected then this option is disabled.

Show Hidden Signals: Lets user choose from a dialog box which hidden signals to show.

Hide Selected Parameter: Hides selected parameter. If no parameters are selected then this option is disabled.

Show Hidden Parameters: Lets user choose from a dialog box which hidden parameters to show. If no parameters are hidden, this menu item is disabled.

Hide Selected Parameter Data: Hides the selected parameter in the parameter window. Hidden parameters are not shown on the screen but they are still continue to function normally.

Show Hidden Parameter Data: Lets user choose from a dialog box which hidden parameters to show. If no parameters are hidden, this menu item is disabled.

Filter Signals: Allows you to select a group of signals based upon a pattern matching their name and to mark the signals as hidden or shown.

Filter Parameters: Allows you to select a group of signals based upon a pattern matching their name and to mark the signals as hidden or shown.

Compare: Opens a sub-menu that allows you to compare all signals or to search for a specific difference between the signals (WaveFormer, VeriLogger and TestBencher Pro feature).

Options Menu

Display Unit [unit]: Menu title shows the display time unit. The submenu allows you to choose a different display time unit.

Base Time Unit [unit]: This allows the base time unit to be set using the base time unit dialog box. **Note: Formula constants**, denoted by a single quote, will not be affected by changes in the base time unit. Default for the base time unit is picoseconds.

Rich Text Support: Turns on and off the rich text edit support. Rich Edits are the edit boxes that let you bold and superscript text. Default is on.

Status Bar Messages: displays the function of buttons, menu items and special mode instructions. Default is on.

Grid Settings: Opens the *Edit Text and Edge Grids* dialog which controls the text and edge alignment grids.

Design Preferences: Affects how functions work in the program.

Drawing Preferences: Affects how objects appear in the Diagram window.

Simulation Preferences: Controls how the interactive HDL simulator works and the default settings for the register and latch models.

Test Bench Preferences: Controls how test benches are created by TestBencher Pro including: the test bench language, transition timing, and the default settings for signals and sample parameters.

Diagram-Level Test Bench Settings: Allows use files, time out settings and HDL code enable toggling for specific diagram components for a specific diagram (TestBencher Pro feature).

Parameter Window Preferences: Controls the display of the min/max columns in the parameter window. These columns can display either formula or the numerical value of the formula.

Text/Color Preferences: Opens the *Text and Color Preferences* dialog that allows you to determine local and global Font and Color settings.

Report Menu

Open Report Tab: Opens a new file in the Report window (*Chapter 12: Report Window*).

Close Report Tab: Closes the active file in the report window.

Save Report Tab: Saves the active file in the Report window.

Save Report Tab As: Saves the active file under a different name.

Print Report Tab: Prints the active file.

Window Menu

Show Toolbars: Turns dockable toolbars on and off.

Cascade: Layers the various windows with the active window on top.

Tile Horizontally: Tiles the windows in a "landscape" fashion (top/bottom).

Tile Vertically: Tiles the windows in "portrait" fashion (side by side).

Redraw: Forces that window to completely redraw.

Report *filename*: Displays the name of the file currently open in the Report window. Check it to maximize the window and bring it to the top.

Parameter *diagram file*: If checked, this will maximize the Parameter window and bring it to the top.

Diagram *diagram file*: If checked, this will maximize the Diagram window and bring it to the top.

Project *project file*: If checked, this will maximize the project window and bring it to the top (available with VeriLogger and TestBencher pro).

Appendix B: Colors and their Meaning

Red indicates activated buttons or error conditions.

Red Buttons are activated buttons. There are two types of buttons that stay red when activated. **Mode Buttons** control the functionality of the right mouse button (delay, hold, setup, sample, text, and marker). **State Buttons** control the level of the next signal to be drawn.

Red Delays are delays that cannot draw themselves properly. Look for an error condition in the parameter window to correct the error.

Red Setups and Holds are constraints whose margins are violated. Look for delays and signal transitions that have been moved too close to the control signal of the constraint for the cause of the violation.

Red Signals are signals that could not be simulated because there was an error in the simulation or because the signals were removed from the simulation using the "Simulate Independently" feature.

Red Transitions (rectangles or vertical bars) in the Diagram window are caused by edge transitions that do not change state (i.e. high to high). To fix the error, either delete the transition or change one of the two segment states to a different state. To delete all the red transitions choose the **Edit > Clear Red Events**.

Red Clocks are clocks whose formulas or values are invalid so that they can no longer draw themselves properly. Double click on a clock segment to open the *Edit Clock Properties* dialog box.

Blue objects are used for measurement and information.

Blue delta button displays the time between the cursor and the blue delta triangle on the timeline. See delta button.

Blue delta triangle is a marker left behind each time the left mouse button is released in the Diagram window. See delta button.

Blue Signals are signals that are defined to be inputs to a test bench (signals that the simulated circuit generates and a test bench generally verifies for correct performance).

Blue signal names with a blue waveform indicate simulated signals (See *Chapter 12: Interactive HDL Simulation*).

Blue Delays are delays that set only the min edge of the delayed transition. Either the delay is a min-only delay or another delay is dominant at the delayed transition (for more information, see multiple delays).

Blue signal names with a black waveform indicates a comparison signal that did not find any differences to report. (See *Chapter 16: Waveform Comparisons*)

Light Gray indicates a selected object or uncertainty.

A **Gray dotted box** shows a selected object in the Diagram window like a segment, text object, or a parameter. Select an object by left clicking on it.

Light Gray signal transitions are the uncertain areas in which a signal transition may change state. An uncertainty region is generated by delays whose min and max times are not equal.

Gray Delays are delays that do not set either edge of the delayed transition. Either the min/max values are blank and can be set by double clicking on the delay. Or another delay is dominant at the delayed transition.

Dark Gray waveforms indicate an invalid waveform state where the value of the signal is unknown.

Green indicates selection.

Green Selection Bar shows a selected signal transition.

Green Delays are delays that set only the max edge of the delayed transition. Either the delay is a max-only delay or another delay is dominant at the delayed transition (for more information multiple delays).

Appendix C: Common Questions

Question: What is the difference between TestBencher Pro, WaveFormer Pro, and Timing Diagrammer Pro?

Answer: Each product adds a new capability. Timing Diagrammer is a timing diagram editor. WaveFormer adds waveform stimulus generation, waveform import, and Interactive HDL simulation. TestBencher Pro adds multi-diagram, self-testing test bench and bus-functional model generation for VHDL and Verilog. For a more detailed explanation read the **Differences section** in on-line help.

Question: How do I get my timing diagrams into a word processor?

Answer: There are 6 ways to embed your timing diagrams into a word processor: copy-to-clipboard bitmaps, WMF metafiles, CGM metafiles, Enhanced metafiles, MIF files, and EPS files. Chapter 8 has the instructions for generating the different images. The file **image.doc** located on the distribution diskette has more detail and hints on EPS and MIF files, and explains the differences between the metafile formats.

Question: When I print names of parameters or signals they do not show up on my printout. Why does this happen?

Answer: If you have white or a light color chosen as the font color your black-and-white printer may not be able to represent it on white paper. To reset all the colors for printing choose the **Options\Text Color Preferences\Restore Default fonts and colors** or alter just the color of the object that did not print.

Question: How can I output to a postscript file?

Answer: Your Windows print driver determines the output format generated by a print operation. Install a postscript print driver using the Window's Control Panel program if you don't already have one installed. To create a postscript file, print to a postscript printer with the "Print to EPS file" option checked. Also, type in a file name in the Filename box.

Question: Does WaveFormer Pro run on a Power PC or a Macintosh computer with Windows software emulation?

Answer: Yes, but WaveFormer Pro requires at least a two button mouse to be used efficiently. You can emulate the right mouse button by using the F12 key, or typing Apple Key *. If you have problems call us or your Apple dealer for more suggestions.

Question: The flicker from the time measure buttons or the status bar is bothersome on my monitor, can I turn those off?

Answer: Yes, you can turn either one off. In each window, choose **Options > Status Bar** to turn off the status bar in that window. If you are just learning to use the program, the status bar gives a lot of hints and instructions, so you may want to keep it on until you feel comfortable with the program.

To turn off the measurement buttons, choose **Options > Drawing Preferences** menu. This will open up the *Drawing Preferences* dialog, uncheck the **Continuous Measurement On** checkbox. Now when ever you left click in the Diagram window the black time button will display the mouse position and the blue button will read zero time units.

Appendix D: License Agreement

SynaptiCAD

TestBencher Pro - DataSheet Pro - WaveFormer Pro - Timing Diagrammer Pro - VeriLogger Pro

Software License Agreement

- Read Before Use -

Please read and understand this license.

Note: Throughout this agreement, the word Software refers to the software product that you have licensed from SynaptiCAD.

You have purchased a license to use one of the following products: **TestBencher Pro, DataSheet Pro, WaveFormer Pro**, **VeriLogger Pro**, or **Timing Diagrammer Pro** software. The software is owned and remains the property of SynaptiCAD, is protected by international copyrights, and is transferred to the original purchaser and any subsequent owner of the Software media for their use only on the license terms set forth below. Opening the packaging for **TestBencher Pro, DataSheet Pro, WaveFormer Pro, VeriLogger Pro,** or **Timing Diagrammer Pro** and/or using either **TestBencher Pro, DataSheet Pro, WaveFormer Pro, VeriLogger Pro,** or **Timing Diagrammer Pro** indicates your acceptance of these terms. If you do not agree to all of the terms and conditions, return the unopened Software and manuals immediately for a full refund.

Use of the Software

- **SynaptiCAD** grants the original purchaser ("Licensee") the limited rights to possess and use the Software and User’s Manual ("Software") for its intended purpose. Licensee agrees that the Software will be used solely for Licensee’s internal purposes and that the Software will be installed on a single computer only. If the Software is installed on a networked system, or on a computer connected to a file server or other system that physically allows shared access to the Software, Licensee agrees to provide technical or procedural methods to prevent use of the Software by more than one user.
- One machine-readable copy of the Software may be made for BACKUP PURPOSES ONLY, and the copy shall display all proprietary notices, and be labeled externally to show that the backup copy is the property of SynaptiCAD, and that use is subject to this License.
- Use of the Software by any department, agency or other entity of the U.S. Federal Government is limited by the terms of the below "Rider for Governmental Entity Users."
- Licensee may transfer its rights under this License, provided that the party to whom such rights are transferred agrees to the terms and conditions of this License, and written notice is provided to SynaptiCAD. Upon such transfer, Licensee must transfer or destroy all copies of the Software.

- Except as expressly provided in this License, Licensee may not modify, reverse engineer, decompile, disassemble, distribute, sub-license, sell, rent, lease, give or in any other way transfer, by any means or in any medium, including telecommunications, the Software. Licensee will use its best efforts and take all reasonable steps to protect the Software from unauthorized use, copying or dissemination, and will maintain all proprietary notices intact.

LIMITED WARRANTY **SynaptiCAD** warrants the Software media to be free of defects in workmanship for a period of ninety days from the purchase. During this period, SynaptiCAD will replace at no cost any such media returned to SynaptiCAD, postage prepaid. This service is SynaptiCAD's sole liability under this warranty.

DISCLAIMER LICENSE FEES FOR THE SOFTWARE DO NOT INCLUDE ANY CONSIDERATION FOR ASSUMPTION OF RISK BY SYNAPTICAD, AND SYNAPTICAD DISCLAIMS ANY AND ALL LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR OPERATION OR INABILITY TO USE THE SOFTWARE, EVEN IF ANY OF THESE PARTIES HAVE BEEN ADVISED OF THE POSSIBILITIES OF SUCH DAMAGES. FURTHERMORE, LICENSEE INDEMNIFIES AND AGREES TO HOLD SYNAPTICAD HARMLESS FROM SUCH CLAIMS. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY THE LICENSEE. THE WARRANTIES EXPRESSED IN THIS LICENSE ARE THE ONLY WARRANTIES MADE BY SYNAPTICAD AND ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. THIS WARRANTY GIVES YOU SPECIFIED LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF WARRANTIES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Term

- This license is effective as of the time Licensee receives the Software, and shall continue in effect until Licensee ceases all use of the Software and returns or destroys all copies thereof, or until automatically terminated upon failure of Licensee to comply with any of the terms of this License.

General

- This License is the complete and exclusive statement of the parties' agreement. Should any provision of this license be held to be invalid by any court of competent jurisdiction, that provision will be enforced to the extent permissible, and the remainder of the License shall nonetheless remain in full force and effect. This License shall be controlled by the laws of the State of Virginia, and the United States of America.

Rider For U.S. Governmental Entity Users

This is a rider to **TestBencher Pro/ DataSheet Pro /VeriLogger Pro/ WaveFormer Pro/ Timing Diagrammer Pro** SOFTWARE LICENSE AGREEMENT ("License") and shall take precedence over the License where a conflict occurs.

1. The Software was: developed at private expense (no portion was developed with government funds) and is a trade secret of SynaptiCAD and its licensor for all purposes of the Freedom of Information Act; is "commercial computer software" subject to limited utilization as provided in any contract between the vendor and the government entity; and in all respects is proprietary data belonging solely to SynaptiCAD and its licensor.
2. For units of the DoD, the Software is sold only with "Restricted Rights" as that term is defined in the DoD Supplement to DFAR 252.227-7013 (b)(3)(ii), and use, duplication or disclosure is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Manufacturer: SynaptiCAD, PO Box 10608, Blacksburg, Va 24062-0608 USA.
3. If the Software was acquired under a GSA Schedule, the Government has agreed to refrain from changing or removing any insignia or lettering from the Software or Documentation or from producing copies of manuals or disks (except for backup purposes) and: (1) Title to and ownership of the Software and Documentation and any reproductions thereof shall remain with SynaptiCAD and its licensor; (2) use of the Software shall be limited to the facility for which it is acquired; and (3) if the use of the Software is discontinued at the original installation location and the Government wishes to use it at another location, it may do so by giving prior written notice to SynaptiCAD, specifying the new location site and class of computer.
4. Government personnel using the Software, other than under the DoD contract or GSA Schedule, are hereby on notice that use of the Software is subject to restrictions that are the same or similar to those specified above.

Appendix E: Standard Parts and X86 Libraries

This Appendix includes the Standard Part and X86 Library instructions, and your library license agreement.

Standard Parts Libraries

The Standard Part Libraries contain over 14,000 timing parameters for commonly used digital parts. The Standard Part Libraries provide you with two major benefits. First they save you the time you would normally spend creating these libraries manually. And second, because similar naming conventions were used for parts from different libraries it is easy to assess the impact of changing part vendors and logic families in your designs.

Directory Structure: The following is the library directory structure that we use at SynaptiCAD, but our library features are setup so that you may choose any directory structure you want as long as files with the same name remain in separate directories.

```
C:\waveform\   /* Directory where WaveFormer Pro is installed */
                /* If you are running Timing Diagrammer Pro then the default directory is
  timing *
/* Part vendor directories */
C:\waveform\lib\mot        /* Motorola */
C:\waveform\lib\nat        /* National Semiconductor */
C:\waveform\lib\ti         /* Texas Instruments */
```

Library Data: Here is a list of our libraries and the sources from which the information was drawn. Please contact us if you find errors or omissions.

c:\waveform\lib\nat\
National Semiconductor FACT Advanced CMOS Logic Databook (400022)

File Name	Part Type	Parm Specs.
3ac.txt	74 AC	3V TA=25C
ac.txt	74 AC	5V TA=25C
act.txt	74 ACT	5V TA=25C

c:\waveform\lib\ti\

Texas Instruments Advanced CMOS Logic Databook 1993 (SCAD001C)

File Name	Part Type	Parm Specs.	Notes
3ac.txt	74/54AC 11000 series	3.3V TA=25C	when no 25C available, 74 data was used
ac.txt	74/54AC 11000 series	5.0V TA=25C	when no 25C available, 74 data was used
act.txt	74/54 ACT 11000 series	TA=25C	
3ac16.txt	74/54AC 16000 series	3.3V TA=25C	when no 25C available, 74 data was used
ac16.txt	74/54AC 16000 series	5.0V TA=25C	when no 25C available, 74 data was used
act16.txt	74/54 ACT 16000 series	TA=25C	

Texas Instruments ALS/AS Logic Data Book 1986 (SDAD001B)

File Name	Part Type
74als.txt	74 ALS
54als.txt	54 ALS
74as.txt	74 AS
54as.txt	54 AS

Texas Instruments High-Speed CMOS Logic Databook 1988 (SCLD001B)

File Name	Part Type	Parm Specs.
74hc.txt	74 HC series	4.5V CL=50pf
74hc150.txt	74 HC series	4.5V CL=150pf
54hc.txt	54 HC series	4.5V CL=50pf
54hc150.txt	54 HC series	4.5V CL=150pf
74hct.txt	74 HCT series	4.5V CL=50pf
74hct150.txt	74 HCT series	4.5V CL=150pf
54hct.txt	54 HCT series	4.5V CL=50pf
54hct150.txt	54 HCT series	4.5V CL=150pf

Texas Instruments TTL Logic Data Book 1988 (SDLD001A)

File Name	Part Type
ttl.txt	74/54 TTL
s.txt	74/54 S
ls.txt	74/54 LS

Texas Instruments F Logic (SN54/74F) 1994 (SDFD001B)

File Name	Part Type	Parm Spec.
f.txt	74/54 F	TA=25C

c:\waveform\lib\mot\
Motorola MECL Device Data (DL122 rev3)

File Name	Part Type	Notes
mc10H.txt	MECL 10KH	MC10H000 series
mc10.txt	MECL 10K	MC10000 series
mc16.txt	MECL III	MC1600 series

X86 Microprocessor Libraries

The X86 Microprocessor Libraries contain over 900 timing parameters for the Intel486 and Pentium processors (covers the timing for 18 different Intel486 processors and 6 different Pentium processors). The X86 Libraries provide you two major benefits. First they save you the time you would normally spend creating these libraries manually. And second, because similar naming conventions were used for parts from different libraries it is easy to assess the impact of changing the speed or power grade of the processors.

Library Naming Conventions

The timing parameter names contain the exact Intel timing parameter name used in the data book and a special SynaptiCAD postfix like:

IntelParameterName_SynaptiCADPostfix

The SynaptiCAD postfix indicates how the parameter should be implemented inside WaveFormer Pro or Timing Diagrammer Pro. The following table shows the possible postfix endings and how to model the parameters in WaveFormer Pro or Timing Diagrammer Pro.

Postfix	What It means	How to Model It
tw	pulse width	Use either a Setup or Hold parameter to monitor the size of the pulse width. Or use a Delay to force the pulse width to be the exact size indicated by the values.
tvp	valid propagation delay	Use a Delay parameter.
tfp	float propagation delay	Use a Delay parameter.
ts	setup time	Use a Setup parameter.
th	hold time	Use a Hold parameter.
tjHL or tjLH	fall time of an edgerise time of an edge(sometimes called edge jitter)	Use these timing values in the jitter formulas of clocks and in the min and max formulas of delays. There is no single parameter that specifies the rise or fall times of an edge. This is modeled indirectly by the gray uncertainty regions on the edges of signals and clocks.
any "a"	means asynchronous	just a hint to meaning of parameter
any "H" or "L"	means High or Low	just a hint to meaning of parameter

486 Libraries

c:\waveform\lib\x86\ /* if you are using Timing Diagrammer Pro then c:\timing\...*/
Intel databook "Intel486 Microprocessor Family 1994", (order number 242202-001)

file name	processor	databook reference
20B3V.txt	40MHz Intel DX2(20MHz max) 3.3v 40MHz WriteBackEnhanced Intel DX2 (20MHz)3.3v	tables 17-17 & 17-21
25B3V.txt	25MHz Intel 486 SX 3.3v 50MHz Intel DX2 (25MHz Max) 3.3v 50MHz WriteBackEnhanced Intel DX2 (25MHz) 3.3v	tables 17-17 & 17-21
33B3V.txt	33MHz Intel 486 SX 3.3v 33MHz Intel 486 DX 3.3v	tables 17-17 & 17-21
75M_DX4.txt	75MHz Intel DX4 (25MHz Bus) 3.3v	tables 17-18 &17-22

file name	processor	databook reference
83M_DX4.txt	83MHz Intel DX4 (33MHz Bus) 3.3v	tables 17-19 &17-22
100M_DX4.txt	100MHz Intel DX4 (50MHz Bus) 3.3v	tables 17-20 &17-22
25B5V.txt	25MHz Intel 486 SX 5v	tables 17-23 & 17-26
33B5V.txt	33MHz Intel 486 SX 5v 33MHz Intel 486 DX 5v	tables 17-23 & 17-26
25B5V_X2.txt	50MHz Intel SX2(25MHz max) 5v 50MHz Intel DX2(25MHz max) 5v 50MHz WriteBackEnhanced Intel DX2 (25MHz) 5v	tables 17-23 & 17-26 Modified by 17-24
33B5V_X2.txt	66MHz Intel 486 DX2 5v 66MHz WriteBackEnchanced Intel DX2(25MHz) 5v	tables 17-23 & 17-26 Modified by 17-24
50M_DX.txt	50MHz Intel 486 DX 5v	tables 17-25 & 17-26

486 Library Assumptions:

1) SynaptiCAD assumed that the minimum rise and fall times of a signal is 0ns, if Intel did not specify a minimum time. See parameters with postfix endings _tjHL and _tjLH.

2) SynaptiCAD assumed that the minimum float delay time is 1ns, if Intel did not specify a minimum time. See parameters with postfix endings _tfp.

Pentium Libraries

c:\waveform\lib\x86\ /* if you are using Timing Diagrammer Pro then c:\timing\...*/
Intel databook "Pentium Processors and Related Products 1995", (order number 241732-002)

file name	processor	databook reference
P66.txt	66Mhz Pentium, 5v	table 3-3, page 2-21
P60.txt	60Mhz Pentium, 5v	table 3-4, page 2-24
P75_B50.txt	75Mhz Pentium, 50Mhz Bus, 3.3v	table 12, page 2-57

file name	processor	databook reference
P100_B50.txt	100MHz Pentium, 50MHz Bus, 3.3v maybe 75MHz/50Mhz see note 3	table 12, page 2-108
P90_B60.txt	90MHz Pentium, 60MHz Bus, 3.3v	table 14, page 2-114 and table 15, page 2-118
P100_B66.txt	100MHz Pentium, 66MHz Bus, 3.3v	table 16, page 2-119 and table 17, page 2-123

Pentium Library Assumptions:

1) SynaptiCAD assumed that the minimum rise and fall times of a signal is 0ns, if Intel did not specify a minimum time. See parameters with postfix endings _tjHL and _tjLH.

2) SynaptiCAD assumed that the minimum float delay time is 1ns, if Intel did not specify a minimum time. See parameters with postfix endings _tfp.

3) Table 12, page 2-108, of the Intel data book claims to be valid timing for both the 100MHz and 75MHz Pentiums with 50MHz buses. However this table has different timing values than the table 12, page 2-57 which is also timing for a 75MHz Pentium with a 50MHz Bus.

Hints on Using Standard Part Libraries

Adjust Display Time Units and Base Time Units before loading the Libraries. All of the libraries in the Standard Parts Libraries are in units of nanoseconds, therefore the Display Time Unit must be ns (nanoseconds). To get maximum resolution we recommend that the Base Time Unit = ps (picoseconds), that way 5.5 ns doesn't get rounded to 6 ns. The Display Time Unit must be set before the libraries are loaded.

Do the Parameter Libraries tutorial included in this manual. The tutorial explains how to add libraries to a project's library list and how to reference parts from inside the libraries.

Pulse Width parameters can be used in your timing diagrams in one of two ways. First if you want to monitor the width of a pulse and be warned if the pulse becomes two small, then add a setup between the last and the first edge of the signal pulse. The setup min value should reference the library parameter containing the width value. The setup max value in most circumstances should be blank.

Second, If you would like to force a pulse to be a specific width then add a delay between the edges of the pulses. The delay's max and min column should reference the Library_part_name.min that contains the pulse width. The .min ensures that the max column uses the min value of the Library part.

Use the F3 Key. Read the parameter library tutorial or the on-line help to learn what the F3 key does. This is the key to effectively using the Parameter Libraries.

Help Topics

When in doubt, read on-line help. To find all the library topics:

- Choose **Help > Table of Contents** menu option in the timing window.
- Click on **Support Functions Index** jump.
- All the library topics are listed in the order in which they should be read. Left click on a topic to read it. Then use the **Back** button to return the previous page.

Part Libraries Software License Agreement

- READ BEFORE USE-

Please read and understand this license.

You have purchased a license to use **SynaptiCAD Part Libraries** software. The software is owned and remains the property of **SynaptiCAD**, is protected by international copyrights, and is transferred to the original purchaser and any subsequent owner of the Software media for their use only on the license terms set forth below. Opening the packaging and/or using SynaptiCAD Part Libraries indicates your acceptance of these terms. If you do not agree to all of the terms and conditions, return the unopened Software immediately for a full refund.

Use of the Software

• SynaptiCAD grants the original purchaser ("Licensee") the limited rights to possess and use SynaptiCAD Part Libraries ("SP Libraries") for their intended purpose. The SP Libraries are licensed to be used with a specific copy of Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro, or TestBencher Pro. The SP Libraries licensed to one copy of Timing Diagrammer Pro, WaveFormer Pro, VeriLogger pro, or TestBencher Pro may not be used by another copy of Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro, or TestBencher Pro even if the properly registered version is not being used at that time. Licensee agrees that the Software will be used solely for Licensee's internal purposes and that the Software will be installed on a single computer only.

• One machine-readable copy of the Software may be made for BACKUP PURPOSES ONLY, and the copy shall display all proprietary notices, and be labeled externally to show that the backup copy is the property of SynaptiCAD, and that use is subject to this License.

• Use of the Software by any department, agency or other entity of the U.S. Federal Government is limited by the terms of the below "Rider for Governmental Entity Users."

• Licensee may transfer its rights under this License, provided that the party to whom such rights are transferred agrees to the terms and conditions of this License, and written notice is provided to SynaptiCAD. Upon such transfer, Licensee must transfer or destroy all copies of the Software.

• The Licensee is allowed to edit and add new entries to the SP Libraries. The modified software remains the property of SynaptiCAD and is protected under the same restrictions as the original software.

LIMITED WARRANTY SynaptiCAD warrants the Software media to be free of defects in workmanship for a period of ninety days from the purchase. During this period, SynaptiCAD will replace at no cost any such media returned to SynaptiCAD, postage prepaid. This service is SynaptiCAD’s sole liability under this warranty.

DISCLAIMER LICENSE FEES FOR THE SOFTWARE DO NOT INCLUDE ANY CONSIDERATION FOR ASSUMPTION OF RISK BY SYNAPTICAD, AND SYNAPTICAD DISCLAIMS ANY AND ALL LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR OPERATION OR INABILITY TO USE THE SOFTWARE, EVEN IF ANY OF THESE PARTIES HAVE BEEN ADVISED OF THE POSSIBILITIES OF SUCH DAMAGES. FURTHERMORE, LICENSEE INDEMNIFIES AND AGREES TO HOLD SYNAPTICAD HARMLESS FROM SUCH CLAIMS. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY THE LICENSEE. THE WARRANTIES EXPRESSED IN THIS LICENSE ARE THE ONLY WARRANTIES MADE BY SYNAPTICAD AND ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. THIS WARRANTY GIVES YOU SPECIFIED LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF WARRANTIES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Term

• This license is effective as of the time Licensee receives the Software, and shall continue in effect until Licensee ceases all use of the Software and returns or destroys all copies thereof, or until automatically terminated upon failure of Licensee to comply with any of the terms of this License.

General

• This License is the complete and exclusive statement of the parties’ agreement. Should any provision of this license be held to be invalid by any court of competent jurisdiction, that provision will be enforced to the extent permissible, and the remainder of the License shall nonetheless remain in full force and effect. This License shall be controlled by the laws of the State of Virginia, and the United States of America.

Rider For U.S. Governmental Entity Users

This is a rider to the SynaptiCAD Part Libraries SOFTWARE LICENSE AGREEMENT ("License") and shall take precedence over the License where a conflict occurs.

1. For units of the DoD, the Software is sold only with "Restricted Rights" as that term is defined in the DoD Supplement to DFAR 252.227-7013 (b)(3)(ii), and use, duplication or disclosure is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Manufacturer: SynaptiCAD, PO Box 10608, Blacksburg, Va 24062-0608 USA.

2. If the Software was acquired under a GSA Schedule, the Government has agreed to refrain from changing or removing any insignia or lettering from the Software or Documentation or from producing copies of manuals or disks (except for backup purposes) and: (1) Title to and ownership of the Software and Documentation and any reproductions thereof shall remain with SynaptiCAD and its licensor; (2) use of the Software shall be limited to the facility for which it is acquired; and (3) if the use of the Software is discontinued at the original installation location and the Government wishes to use it at another location, it may do so by giving prior written notice to SynaptiCAD, specifying the new location site and class of computer.

3. Government personnel using the Software, other than under the DoD contract or GSA Schedule, are hereby on notice that use of the Software is subject to restrictions that are the same or similar to those specified above.

Tutorial Overview

There are several tutorials shipped with all versions of Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro, and TestBencher Pro. The first four tutorials are printed here for your convenience. However, SynaptiCAD's base timing analysis package is being constantly improved and expanded, therefore we may have added new tutorials or changed the existing ones since the printing of this manual. To find the latest copies of the tutorials read the **Tutorials** page in the on-line help. This will help you print out any new tutorials or indicate any major changes.

- **Basic Drawing and Timing Analysis**, covers the basic timing diagram editing environment. How to set up the base time unit and the display time unit of a timing diagram. How to draw and edit signals, delays, and setups. How to make basic time measurements. This tutorial is essential to anyone who is evaluating or learning to use SynaptiCAD products.

- **Simulation, Waveform Generation, and Parameters**, covers the techniques for editing and customizing the information displayed by a parameter and its position on the screen. It also covers HDL simulation and Temporal waveform generation of signals, virtual and group buses, and general signal manipulation. This tutorial teaches the techniques to help quickly generate timing diagrams without having to spend a lot of time unnecessarily drawing signals. This tutorial is essential to anyone evaluating SynaptiCAD products.

- **Parameter Library Use and Setup**, covers the basic concepts of parameter libraries and the use of macro lists. This tutorial is important to do before starting a large project. Using parameter libraries and macro lists can save you a lot of time if they are configured correctly.

- **HDL Stimulus Generation,** covers the basic concepts of VHDL-Verilog stimulus generation, language independent hex and binary bus translation, default mapping modes, and user defined types. WaveFormer, VeriLogger and TestBencher users should do these tutorials.

- **Advanced Simulation** (on-line tutorial), demonstrates how WaveFormer Pro, VeriLogger, or TestBencher Pro can quickly simulate a digital system of moderate complexity. You will model state machines using Boolean equations, use the Report window to find simulation errors, enter direct HDL code, model tristate gates, model n-bit gates, and call external HDL models. WaveFormer Pro, VeriLogger Pro and TestBencher Pro users should do this tutorial.

- **Basic Verilog Simulation** (on-line Tutorial), covers the basic simulation features of VeriLogger Pro. This tutorial also discusses how to work create and manage projects, as well as how to build and simulate your design. VeriLogger Pro and TestBencher Pro users should perform this tutorial.

- **TestBencher Pro: Basic Tutorial** (on-line tutorial), covers the basic concepts of using TestBencher Pro to generate bus-functional models for Verilog and VHDL. It covers signal properties (type, direction, vector size, and bi-directional segments), samples, parameterized state values, end diagram markers, interface diagrams, modifying top-level template files, and generating test benches. TestBencher Pro users should do this tutorial.

- **TestBencher Pro: Performing a Sweep Test** (on-line tutorial), covers how to use TestBencher Pro to create a global clock generator, work with cyclic timing transactions, create a sweepable delay, and create a continuous setup checker. TestBencher Pro users should perform this tutorial

Basic Drawing and Timing Analysis

This tutorial demonstrates the basic timing diagram editor features of WaveFormer Pro, Timing Diagrammer Pro, VeriLogger Pro and TestBencher Pro. It teaches you how to draw timing diagrams using delays, setups, clocks and part libraries and how to use timing diagrams to help detect timing errors in digital designs. It also covers the waveform editing features, measurement and quick access buttons.

You will draw the timing diagram for the circuit shown in Figure 1. This circuit divides the clock frequency in half. Both the flip-flop and the inverter have propagation times that delay the arrival of the Dinput signal. If the Dinput is delayed too long it will violate the data-to-clock setup time. This increases the risk of the flip-flop failing to clock in the data and may lead to the flip-flop entering a metastable state.

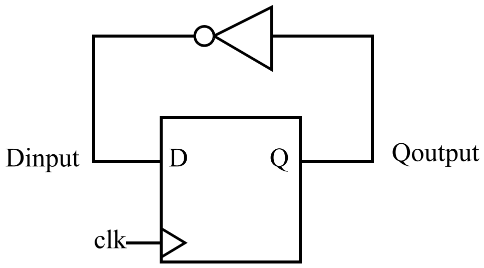


Figure 1: Tutorial circuit

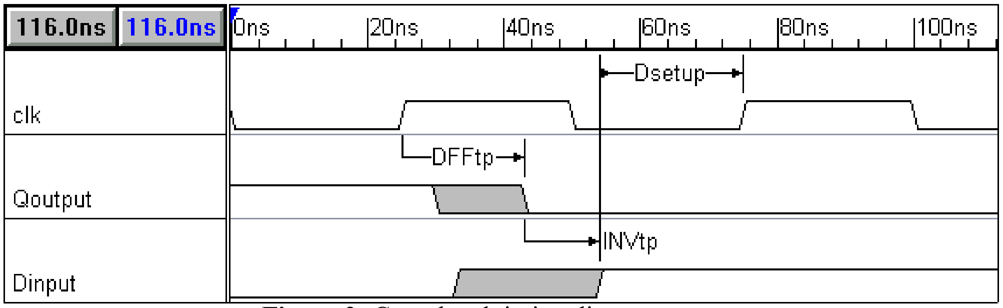


Figure 2: Completed timing diagram

Figure 2 is the completed timing diagram. The first thing you may notice is the gray signal transitions caused by the min/max values of the component delays. The gray areas of the signal transitions are uncertainty regions, which indicate that the signal may transition any time during that period. This is a little disconcerting especially if you have been using a low-end simulator that cannot compute both min and max at the same time. This representation shows the entire range of possible circuit performance. With WaveFormer Pro, there won't be any surprises during production when you get components at extreme ends of their tolerance range.

Circuit Parameters:

clk	20MHz	(50ns period)

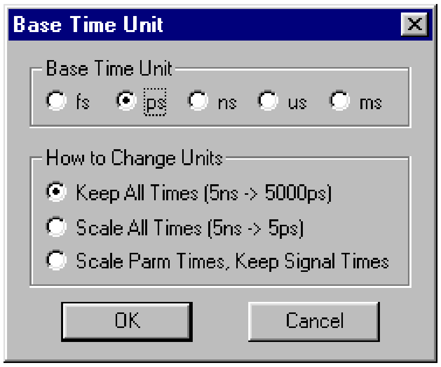
DFFtp	5-18ns	D flip-flop (74ALS74): Clock to Q propagation time
Dsetup	15ns minimum	D flip-flop (74ALS74): D to rising edge Clock setup time
INVtp	3-11ns	Inverter (74ALS04): propagation time

1) Set the Base Time Unit

At the beginning of each project, you will set the base time unit. The base time unit is the smallest representable amount of time that WaveFormer Pro will be able to display. The base time unit determines the range of times that can be represented in your timing diagram. All time values are internally stored in terms of the base time unit.

In the circuit in Figure 1, the propagation times for the gates are in units of nanoseconds and the clock has a period of 20ns. Generally it is a good idea to set the base time unit for your project one unit below the units you are working in for best rounding performance during division operations (clock frequencies are inverted and stored internally as clock periods). Therefore we will set the base time units to picoseconds. To set the base time unit:

1. Select the **Options > Base Time Unit** menu option. This displays the *Base Time Unit* dialog box with radio buttons that set the base time unit. The other options control how any existing parameters or signals are changed when the base time unit is changed and have no effect on an empty timing diagram. See the on-line help if you want to know more about these options.
2. Click on the **ps** radio button to make picoseconds the base time unit (if it is not already selected).
3. Press the **OK** button to close the dialog.



Base Time Unit: fs [✓ ps] ns us ms

How to Change Units: [✓ Keep All Times (5ns -> 5000ps)] Scale All Times (5ns -> 5ps) Scale Parm Times, Keep Signal Times

2) Set the Display Time Unit

Next, you need to set the display time unit. The display time unit sets the units for times which you enter and for times which are displayed. Set the display time unit to the units you most commonly use in the design.

To set the display time unit:

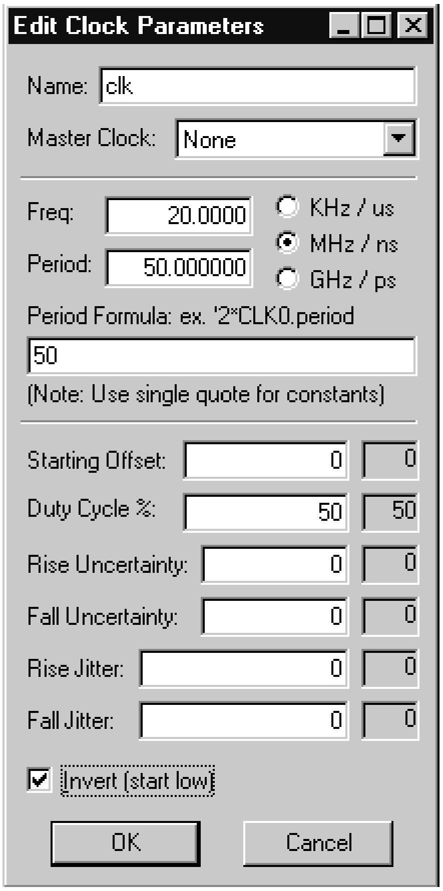
1. Select the **Options > Display Unit** menu option. This will display a submenu of display time units. The checked time is the current display time unit (Default is ns = nanoseconds).
2. Click on **ns**, to make nanoseconds the display time unit if it is not already checked.

3) Add the Clock

First, we will create the clock. The clock is named **clk**, has a period of **50ns** (20MHz), and starts with a low segment.

To add a clock:

1. Move the mouse cursor over the **Add Clock** button Add Clock (DO NOT CLICK), located in the top left hand corner of the Diagram Window.
2. Notice that the status bar at the very bottom of the window reads "Add a clock signal" Add a clock signal . This status bar changes depending on the mouse location and the mode that you are in. Move the mouse around and watch the status bar change. The status bar is very useful when you want to know what buttons do or when you need to know what to do next in the middle of a program operation.
3. Press the **Add Clock** button Add Clock .
4. The *Edit Clock Parameters* dialog box will appear.
5. Enter the name **clk** in the *Name:* edit box.
6. Enter **50** in the *Period:* box. Make sure the **MHz/ns** radio button is selected. Note that the frequency will change to match the new period value, when you move the selection to another control.
7. Check the **invert** check box. Clocks are normally displayed high at time zero, so "invert" makes the clock start low at time zero.
8. Press the **OK** button to close the dialog box.



Edit Clock Parameters

Name: clk

Master Clock: None

Freq: 20.0000

Period: 50.000000

( ) KHz / us [✓ MHz / ns] ( ) GHz / ps

Period Formula: ex. '2*CLK0.period

50

(Note: Use single quote for constants)

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

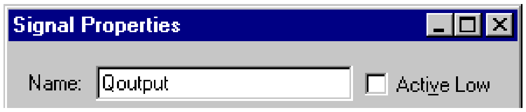
☑ Invert (start low)

Note: For more information on clocks, master clocks, clocks with formulas, and clock grids read *Chapter 2: Clocks* in the manual. If you made a mistake designing the clock, then double left click on the clock segment to reopen the *Edit Clock Parameters* dialog box. Double left clicking on a clock edge opens up the *Edge Properties* dialog box which displays the edge time. You may also reach the *Edit Clock Parameters* dialog box by double clicking on the clock name and choosing the **clock properties** button in the *Signal Properties* dialog.

4) Add Signals

Next, add two signals and name them "Qoutput" and "Dinput".

1. Left click twice on the **Add Signal** button Add Signal to add two signals. The signals will be have default names like SIG0 and SIG1.
2. **Double left click** on the **SIG0** signal name to open the *Signal Properties* dialog.
3. Enter **Qoutput** into the *Name* edit box. (DO NOT CLOSE THE DIALOG)


Signal Properties

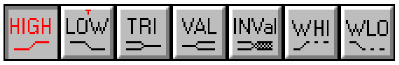
Name: Qoutput ☐ Active Low

4. Press the **Next** button or **ALT-N** to move to the next signal on the list. SIG1 is now displayed in the Name: edit box.
5. Enter **Dinput** into the *Name:* edit box and press the **OK** button to close the dialog.

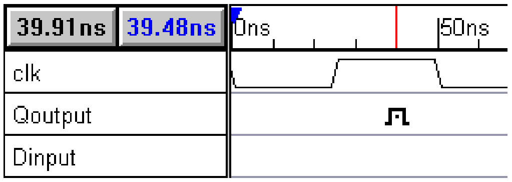
If you accidentally close the *Signal Properties* dialog, double click on the signal name to open the dialog again. The Boolean Equation and Simulation features of the *Signal Properties* dialog are covered in the Simulation, Waveform Generation, and Parameters tutorial. The *Signal Properties* dialog is modeless, so you can leave the dialog open while you perform actions on the timing diagram.

5) Drawing Signal Waveforms

Next, we will draw some random waveforms to become familiar with the drawing environment.

1. Notice the buttons with the waveforms on them.  These are the State Buttons. The active button is colored red and indicates the type of signal state that will be drawn next. In this case, the HIGH signal state is active.

2. **Move** the mouse cursor to inside the Diagram window at the same level as the signal name **Qoutput**, and at about **40ns**.



3. **Left click** to draw a waveform segment from 0ns to the cursor. Notice that a HIGH signal was created.
4. A different state button is now activated. The State Buttons automatically toggle between the two most recently activated states. The small red T above the state name denotes the toggle

state, for instance, LOW . (If you have a 3 button mouse, click the middle mouse button to toggle between the two most recently activated state buttons.)

5. Move the cursor to about **80ns** on the same signal and **left click**. Now a LOW segment is drawn from the end of the HIGH signal to the location of the cursor.
6. Left click on the TRI button to activate the tristate State Button and draw another waveform segment.
7. Draw more segments, using all the states except the HEX button. The HEX state button is used in defining multi-bit signals and signals which have a user defined VHDL type. This button is covered in later tutorials. For the time, experiment with the graphical states.

Your drawing should be a mess, or at least look nothing like Figure 2.

6) Edit signal waveforms

There are four main editing techniques used to modify existing signals (Note: these techniques will not work on clocks). The most commonly used technique is the dragging of signal transitions to adjust their location. The other three techniques all act on signal segments, the waveforms between any two consecutive signal transitions. The segment waveform can be changed, deleted, or a new segment can be inserted within another segment. Use each of the following techniques:

1. **Move a signal transition:** Left click down on a signal transition and drag it to the desired location. A green bar will appear that follows the mouse cursor. Release the mouse button when the green bar is at the location where you wish to place the transition.
2. **Change the state of a segment:** A segment is the waveform between two consecutive signal transitions. Left click on the segment to select it (a selected segment has a highlighted box drawn around it). Then left click on the State Button of the new state.

If you try to select a narrow segment and one of the transitions gets selected, widen the segment by clicking the **Zoom In** button Zoom In , located on the right hand corner of the button bar.

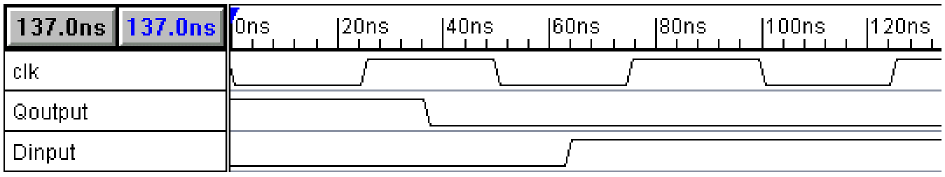
3. **Delete a segment:** Select a segment (see above) and then press the **delete** key on the keyboard.
4. **Insert a segment:** Inside a large segment, Left click down and drag to the right then release. A new segment will be added in the middle of the original segment. For this operation to work the original segment must be wide enough to be selected.

These techniques will not work on clocks. This is because clocks have fixed edges and segments. To edit a clock, double click on a segment of the clock waveform in the Diagram window. This causes the *Edit Clock Parameters* dialog box to appear. All clock parameters can be changed in this dialog box. If you cannot double click on a segment without selecting a transition, zoom in until the segment is large enough.

For more information, read *Chapter 1: Signals and Waveforms*.

7) Adjust your diagram so it resembles the figure below

Now use the above techniques to edit the signals so they have roughly the same transitions as the signals in the figure below. This is not the normal way to create a timing diagram, but it will teach you how to use the editing features of WaveFormer Pro. Make sure you try all the editing techniques.



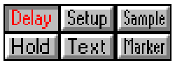
Tile the Diagram and Parameter windows so that you will be able to see the interaction between the two windows. The Report window is not used in this tutorial, so you can minimize it.

- Select one of the **Window > Tile** menu options in the timing window.

Adjust the zoom level of the drawing so that only 3 whole clock periods are shown on the screen.

- Left mouse click the **Zoom In** or **Zoom Out** buttons, which are located on the right hand corner of the button bar, to show less or more of the waveforms.

8) The right mouse button (mode buttons):

In the next sections we will add delays, setups, and comments to the timing diagram. These objects are added using the right mouse button. The function the right mouse button is determined by the second group of buttons on the button bar marked DELAY, HOLD, SETUP, TEXT, SAMPLE and MARKER. The red or active mode button indicates the current functionality of the right mouse button. To activate a different mode button, left click on it. 

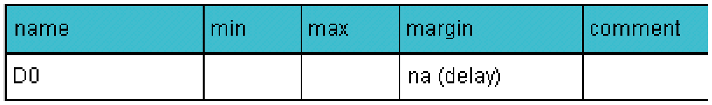
9) Add the D flip-flop propagation delay

Add the delay that represents the propagation time from the positive edge of the clock to the Qoutput of the D flip-flop. To add the delay do the following:

1. Activate the **Delay** mode.

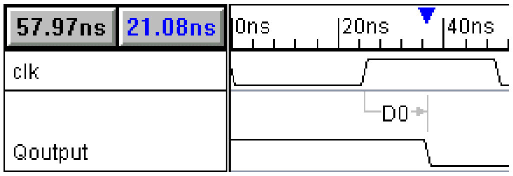
2. Left mouse click on the first rising edge of the clock.
3. **Right** mouse click on the first falling edge of the **Qoutput** signal.

This will draw the D flip-flop delay, and creates a blank delay in the parameter window.



name	min	max	margin	comment
D0			na (delay)	

When delays are added, they are blank and do not enforce any timing restraints. Notice that the delay is drawn with gray colored lines, this indicates that the delay is not forcing either the min or max edges of the Qoutput signal. Now edit the delay's parameters.



1. **Double left click** on the parameter name **D0** in the Parameter window to open the *Parameter Properties* dialog. Adjust the position of the *Parameter Properties* dialog so that you can see the parameter in the Diagram window and at least part of the parameter in the Parameter Window. For simplicity we will call the dialog *Parameter Properties*, even though the name at the top may say *Delay Properties* or *Setup Properties*.

2. Type **5** into the min edit box and press the **TAB** key to move control to the max edit box (leave max blank for now). This enters 5 display time units, or 5ns for this project.

Two things happened when you pressed the TAB key:

- First, the falling edge of the Qoutput signal adjusted itself so that it was 5ns from the clock edge. Measure this for yourself using the time readouts above the signal name window. Left click on the first transition and then move the cursor to the second transition. Notice that the blue readout says 5ns or thereabouts, depending on the zoom level and the base time unit. The delay can also be made to display the exact distance by selecting the **distance** choice of the **Display label** section of the *Parameter Properties* dialog (make sure to return D0 to "Global Default" when you are done experimenting).

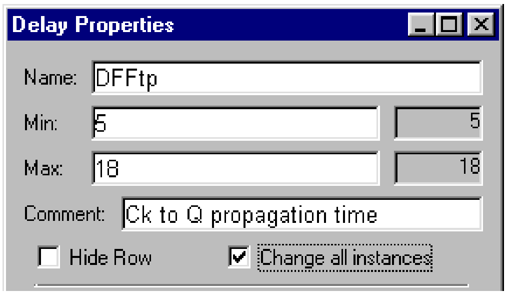
- Second, the delay changed from a gray color to a blue color. Delays are color coded to indicate which delays are forcing the min and max edges of a transition. This type of critical path display is necessary in diagrams where multiple delays drive a single signal transition. The colors are: Gray = none, Blue = Min only, Green = Max only, Black = both min and max. After this tutorial you may want to experiment with the **multdely.tim** file to see the effects of multiple delays on a single transition and critical path color coding.

Next, finish editing the rest of the parameter. Use the *Parameter Properties* dialog that is still open to add the following data:

1. Left mouse click in the **Name** edit box and type **DFFtp** into it.

2. Press **TAB** twice so that the max edit box is selected.

3. Type **18**. This means 18 display time units, or in this project 18ns.
4. **TAB** once so that the comment cell is selected.
5. Enter **Ck to Q propagation time** and press the **Enter** key.



Name: DFFtp
Min: 5 | 5
Max: 18 | 18
Comment: Ck to Q propagation time
☐ Hide Row ☑ Change all instances

Notice that the DFFtp delay is black which indicates that it is forcing both edges of Qoutput. Also notice the falling edge of Qoutput now has a gray uncertainty region. Use the time measure readouts to verify that the edges of the region are 5ns and 18ns from the clock edge (13ns of uncertainty). Double click on the edge to see the exact edge values.

The *Parameter Properties* dialog is **modeless** (other operations can be performed while the dialog is open) and **interactive** (any changes in the dialog fields are reflected in the diagram after you move out of that field). This tutorial has you open and close it several times so that you learn all the different ways to open the dialog. Also, the tutorial attempts to conserve screen area for laptop users. However, in a normal design you will probably want to keep this dialog open much of the time.

Tip: When the *Parameter Properties* dialog is open you can edit a different parameter by double clicking in the diagram or parameter window on the parameter you want to change. If you double click in the diagram window, that instance of the parameter will be edited (the **Change All Instances** checkbox will NOT be checked). If you double click in the parameter window, ALL instances of the parameter will be edited, (the **Change All Instances** checkbox will be checked).

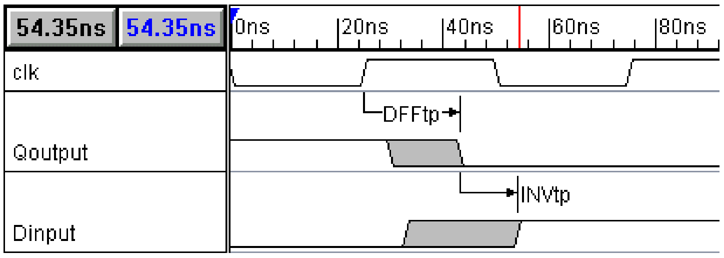
10) Add the Inverter propagation delay

Add the delay that represents the propagation time of the inverter from its input Q to its output D. To add the delay do the following:

1. Activate the DELAY mode by clicking on the **Delay** button.
2. **Left click** on the first falling edge of the **Qoutput** signal (the same edge that ends the "DFFtp" delay).
3. **Right click** on the first rising edge of the **Dinput** signal.

This will draw the inverter delay, and create a blank delay in the Parameter window. Now let's edit the parameters from the inside of the Diagram window instead of going to the Parameter window.

- Double left click on the new delay in the Diagram window and enter the following values in the dialog box that appears:
  - Name is **INVtp**.
  - Min time is **3** ns.
  - Max time is **11** ns.
  - Comment is **Inverter (Q to D) delay**.
  - Click on the **OK** button to close the dialog.

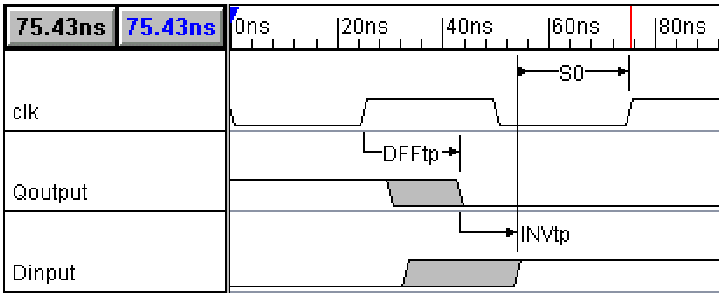


Notice the large uncertainty region for the *Dinput* transition. Click on the first rising edge of **Dinput**, then use the blue delta readout to verify that the uncertainty region lasts for 21ns (13ns from DFFtp + 8ns from INVtp = 21ns). Next, click on the first edge of clk and measure to the end of the uncertainty region of Dinput. If both the inverter and the D flip-flop are slow, Dinput may not transition until 29ns after the clock edge.

11) Add the setup for the Dinput to clock

Next add the setup for Dinput to clock transition.

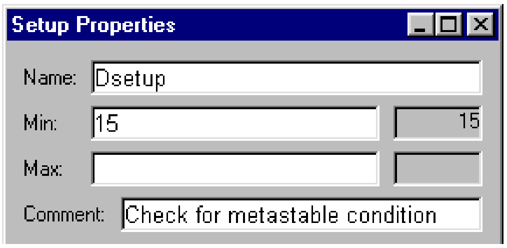
1. Activate the **Setup** mode. Setup
2. **Left click** on the first rising edge of the **Dinput** signal (the same edge that ends the "INVtp" delay).
3. Right click on the second rising edge of the clock.



This will draw the setup parameter. Notice that the arrows of the setup are pointing to the control signal. This means that you added the setup correctly.

Like delays, setups are also created with empty min/max values. They must have a min value before they start to monitor the data signal's position. Now we will edit the setup

1. **Double left click** on the setup name in the Diagram window. This will open the *Parameters Properties* dialog box.
2. Enter **Dsetup** into the Name edit box.
3. Enter **15** into the min edit box.
4. Enter **Check for metastable condition** into the comment edit box.
5. Press the **OK** button to close the dialog.


Setup Properties
Name: Dsetup
Min: 15 | 15
Max:
Comment: Check for metastable condition

Notice that the margin column in the Parameter Window says that there is a **6ns** safety region before the setup is violated. Verify this by clicking on the second rising edge of the clock and placing the cursor on top of the maximum edge of the Dinput signal. The blue time readout should say -21ns (setup time 15ns - measured 21ns = -6ns margin).

Next, we will demonstrate what happens when a setup is violated. Increase the inverter's delay so that the maximum delay is 18ns instead of 11ns:

1. Double left click on **INVtp** in the Diagram window.
2. Type **18** into the **max** edit box and **TAB** to another control.

Notice that the setup has turned red in both the drawing and parameter windows. Change the inverter delay back to **11ns** and click **OK** to close the dialog.

12) Add a free parameter

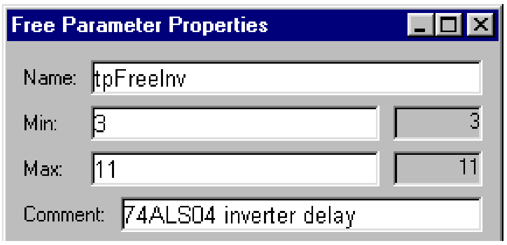
So far we have always directly edited a parameter's values. This is inefficient and error prone if the circuit is large. It would be better to define one variable that held the value and make everything that needed that value reference this variable. Then if the value needs to be changed, you only have to edit one variable.

Free parameters act as variables that can be referenced by other parameters. They are called "free" because these parameters are not attached to any signal transitions in the Diagram window. Let's add a free parameter to hold the propagation times for the inverter.

To add a free parameter:

1. Click the **Add Free Parameter** button Add Free Parameter in the Parameter Window. A blank free parameter is added to the parameter window.

2. Double click on the free parameter to open the *Parameter Properties* dialog box and enter (**tpFreeInv**, **3**ns, **11**ns, and "**74ALS04 inverter delay**") for the (name, min, max, and comment cells) of the free parameter.



Name: tpFreeInv
Min: 3 3
Max: 11 11
Comment: 74ALS04 inverter delay

3. Use the **Previous** and **Next** buttons in the *Parameters Properties* dialog to locate the **INVtp** parameter.
4. Type **tpFreeInv** into the min and max cells of **INVtp**. Changes to the timing values of the free parameter will now affect INVtp.

Note: Free parameters can be saved to special library files which can later be merged into other projects. You can also reference free parameters without including them into your project file by placing libraries in your library search path (**Libraries > Library Preferences** menu option). For more information on free parameters and libraries read *Chapter 10: Parameter Libraries* or perform the *Parameter Libraries* tutorial.

13) Using formulas and constants

Parameters can contain mathematical formulas as well as numeric time values. Legal operations are: multiplication(*), division(/), addition(+), and subtraction(-). For example, the inverter in this circuit could represent three cascaded inverters used to generate a minimum delay of 9ns. To represent this in your timing diagram:

- Enter the following equation into **INVtp**'s **min** edit box: **3 * tpFreeInv**

Free parameter names can also be used with an attributed parameter name such as **tpFreeInv.min** and **tpFreeInv.max**. This gives you the flexibility to specify formulas any way you need. If no attribute is added then a min or max is assumed depending on whether the formula is in the min or max column.

14) Summary

Congratulations! You have completed the *Basic Drawing and Timing Analysis* tutorial. In this tutorial we have covered three main topics: starting a project, drawing and editing signals, and adding and editing parameters.

1) Starting a project

- Always set the **Base Time Unit** one unit below the **Display Time Unit** to avoid rounding errors. Default values are: Base Units= ps and Display Units = ns.

2) Drawing a timing diagram with Signals and Clocks

- Use the **Add Clock** and **Add Signal** buttons to add clocks and signals to the diagram. Double left click on the signal name to edit the signal name.
- Left click to draw a waveform with the state of the selected State Button.

Editing waveforms

- Drag and drop signal transitions.
- To change the graphical state of a segment, select it then press a State Button to indicate the new graphical state.
- To Delete a segment, select it then press the delete key on the keyboard.
- To Insert a segment, left click and drag to the right.

3) Timing Analysis with Parameters

- Add delay, setup, and hold parameters by (1) activating the mode button of that name, (2) left clicking on the first signal transition, and (3) right clicking on the second signal transition.
- Edit parameters by double left clicking on the parameter in either the drawing or the parameter window.
- Free parameters are variables that other parameters reference. Use the **Add Free Parameter** button in Parameter Window to add a free parameter.
- To use a free parameter, type the name of the free parameter in the min or max column. Free parameters can also use the dot min/max property to specify a specific value. For example, Inverter.min retrieves only the minimum value of the parameter named Inverter.

Simulation, Waveform Generation, & Parameters

This tutorial introduces the Interactive HDL Simulation, waveform generation, and parameter display features. This tutorial assumes that you are able to draw signals and can add delays, setups, and holds to those signals. We recommend that beginners start with the *Basic Drawing and Timing Analysis* tutorial, tutbas1.htm, to learn the basics of timing diagram editing, before attempting this tutorial. **Topics Covered:**

1. Interactive HDL Simulation (WaveFormer Pro, VeriLogger Pro and TestBencher Pro only)
2. Advanced Waveform Generation: Temporal Equations & Buses
3. Advanced Parameter Techniques

If you are evaluating Timing Diagrammer Pro and you would like to learn about the simulation features, close the program and restart the evaluation version in WaveFormer Pro mode. Timing Diagrammer Pro customers should go directly to Section 2.

1) Interactive HDL Simulation

WaveFormer Pro, VeriLogger Pro and TestBencher Pro have a built-in Interactive HDL Simulator that can simulate Boolean Equations with delays, registers and latches, and behavioral Verilog code. The simulator is interactive which means changes to input waveforms trigger instant resimulations. This feature greatly reduces the amount of time needed to draw a timing diagram, especially one that models gate level circuits.

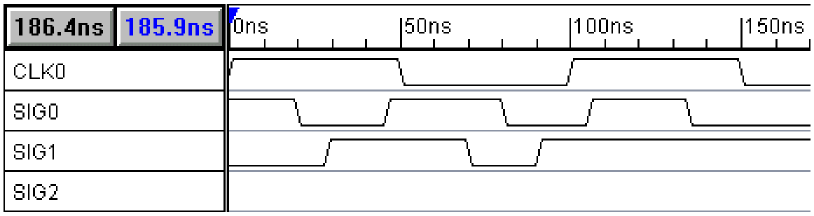


Figure 1.1: Timing diagram used for the basic HDL Simulation tutorial.

Setup a timing diagram for experimenting with the Interactive HDL Simulator:

1. Add a clock named **CLK0**, and accept the default properties of **100ns** period.
2. Add 2 signals, **SIG0** and **SIG1**, to the timing diagram.
3. Draw the waveforms for signals SIG0 and SIG1 so that they resemble the signals in Figure 1.1. These will be the input signals for our simulation.

4. In the Parameter window, click the **Add Free Parameter** button to add a free parameter, **F0**, to the parameter window.

5. **Double left click** on the free parameter **F0** (in the parameter window) to open the *Parameter Properties* dialog and set the min time to **10** and the max time to **15**. Push the **OK** button to close the dialog.
6. Add signal **SIG2** to the timing diagram, but DO NOT DRAW the waveform. We will simulate this signal.

1a) Generate Waveforms From Boolean Equations of other Signals

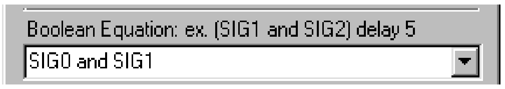
We will begin by simulating a Boolean Equation. WaveFormer Pro accepts Boolean equations in either VHDL, Verilog, or SynaptiCAD's enhanced equation syntax. The SynaptiCAD format supports the following operators: **and** or **&**, **or** or |, **nand**, **nor**, **xor** or ^, **not** or ~ or **!**, and **delay**.

The **delay** operator takes a signal on the left and a time or parameter name on the right and returns a signal. If a parameter name is used on the right hand side of the delay operator, then the equation will simulate true min/max timing. This true min/max timing is the main advantage that SynaptiCAD's format has over the VHDL or Verilog format.

Simulate a Boolean equation:

1. Double left click the **SIG2** signal name to open the *Signal Properties* dialog. Arrange the *Signal Properties* dialog so that you can see the dialog and the 3 signals at the same time. This dialog is modeless, so leave it open for this entire section. <u>All</u> controls and buttons used in this section are contained in the *Signal Properties* dialog.
2. Make sure the Boolean Equation radio button is selected.

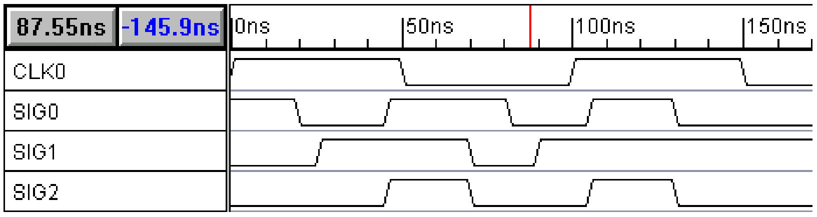
3. Type the following equation into the Boolean equation edit box (signal names are case sensitive): **SIG0 and SIG1**



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

SIG0 and SIG1

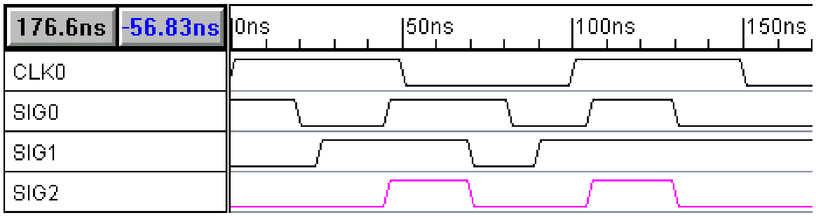
4. Push the **Simulate Once** button and watch the signal draw itself. Notice that SIG2 is the result of the Boolean Equation "SIG0 and SIG1". By default, the **Simulate** radio button is not checked, so if an edge on SIG0 is moved, SIG2 is not automatically resimulated.



Note: If you are using VeriLogger Pro or TestBencher Pro, make sure that the program is in **Auto Run** simulation mode. Debug Run mode will not continuously update signals.

Continuously Simulate the Boolean Equation:

1. Check the **Simulate** radio button. Notice that the SIG2 is now drawn in blue. This color means that the signal is being continuously simulated, and changes in the input waveforms cause automatic resimulations.



2. Move some of the edges on SIG0 and SIG1 and watch SIG2 resimulate. (Notice that you cannot drag and drop SIG2's signal edges because they are calculated edges).

1b) Boolean Equations with Delays

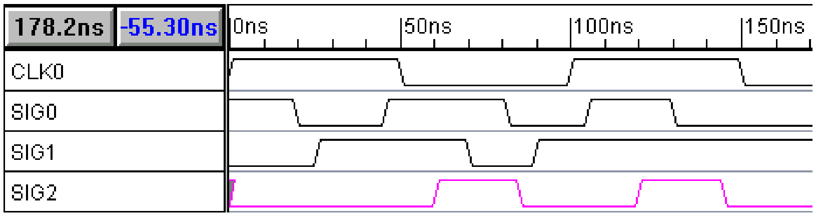
Next we will modify the Boolean to take into account the propagation delay through the AND gate. First we simulate a simple 15ns delay then we will simulate a min/max delay.

Simulate a simple delay:

1. Enter <u>one</u> of the following Verilog, VHDL, or SynaptiCAD equations into the **Boolean equation** edit box of SIG2:

```
#15 (SIG0 & SIG1)
(SIG0 and SIG1) after 15
(SIG0 and SIG1) delay 15
```

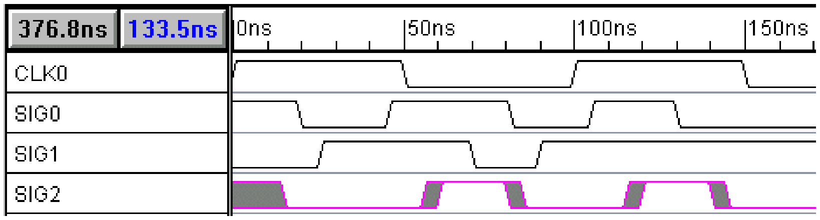
2. Press the **Apply** button and verify that SIG2 is correctly drawn.



Simulate a true min/max delay using SynaptiCAD syntax:

1. Modify the **Boolean Equation** of SIG2 to take into account the min and max propagation delay of the AND gate: **(SIG0 and SIG1) delay F0**. This Boolean equation references the Free parameter *F0* that you added at the beginning of the tutorial.

2. Click the **Apply** button to cause a simulation. Notice the gray uncertainty regions on SIG2. This true min/max timing is the main advantage that SynaptiCAD's format has over the VHDL or Verilog format.



View the HDL code that models the Boolean equation:

1. Click the **Verilog** or **VHDL** radio button to view the HDL code that simulates the Boolean equation. If you wanted to, you could add native HDL code here to perform a special function. Do not modify the code now. The code should resemble the following example:

```
wire #F0min SIG2_wf2 = (SIG0 & SIG1 );
wire #F0max SIG2_wf3 = (SIG0 & SIG1 );
assign SIG2 = (SIG2_wf2 === SIG2_wf3) ? SIG2_wf2 : 'bx;
```

2. Click the **Boolean Equation** radio button to display the Logic Wizard section (or Boolean Equation Section) of the Signal Properties dialog.
3. Leave the *Signal Properties* dialog open. We will be using it in the next section.

Note: This example demonstrated true min/max simulation, however Min-Only and Max-Only simulations can be performed by changing the **Options > Simulation Preferences > Timing Model** drop-down list box.

1c) Registered and Latched Signals

WaveFormer Pro, VeriLogger Pro and TestBencher Pro can register or latch the result of a Boolean equation. Figure 1.2 represents the circuit that is modeled.

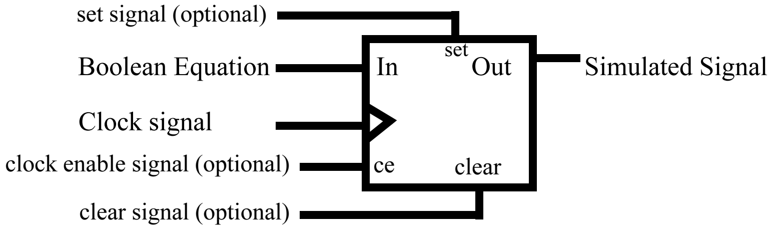
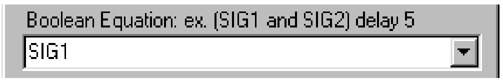


Figure 1.2: Register and Latch circuit modeled by the Logic Wizard section of the Signal Properties dialog

The *Signal Properties* dialog should still be open and displaying the SIG2 information from the last section. Let's experiment with the register and latch functions:

1. Enter the equation **SIG1** into the **Boolean Equation** edit box of **SIG2**.



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

SIG1

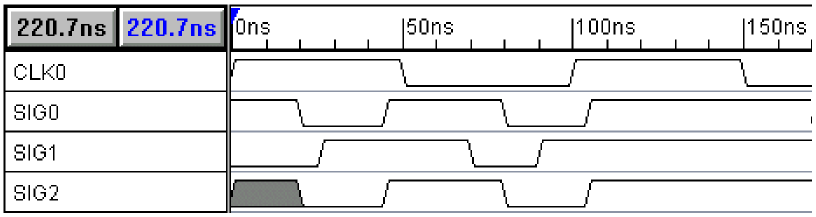
2. Click the **Simulate Once** button to simulate the equation. SIG2 should look like an exact copy of SIG1. When we register SIG2 you can visually compare it to SIG1 to see the effects of the register.
3. Next use the **Clock** drop down list box and choose **SIG0** as the clocking signal. The clocking signal can be any clock or signal in the timing diagram (the default value "Unclocked" means no flip-flop is present).

Clock: SIG0

4. Use the **Edge/Level** drop down list box (on the right side of the dialog) to select **both** as the trigger signal.

Edge/Level: both

5. Click the **Simulate Once** button to simulate the circuit. Notice that SIG2 only transitions when SIG0 has a positive or negative edge transition (move some edges on SIG0 and SIG1 to verify this).



Whether a Register or a Latch is simulated depends on the type of triggering in the **Edge/Level** list box. For a Register circuit choose **neg** for negative edge triggering, **pos** for positive edge triggering, and **both** for edge triggering. For a Latch circuit choose either **low** or **high** level latching.

- Choose different **Edge/Level** values and press the **Simulate Once** button to verify the operation of the register and latch functions.

1d) Set and Clear lines

The **set** and **clear** are useful when defining circuits whose initial value needs to be specified. In this example we will demonstrate how to design a **divide by 2** circuit using a negative edge triggered register with an asynchronous active-low set line. To specify the initial value:

1. Press the **Add Signal** button to create a new signal named SIG3.

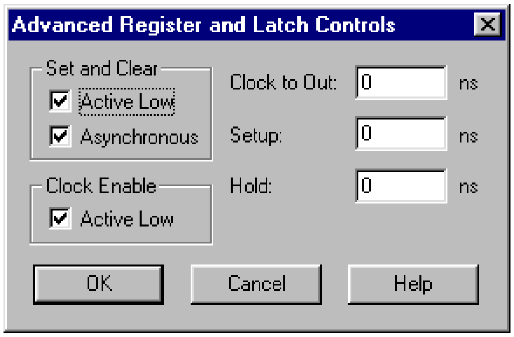
2. Double click on the **SIG3** name to open the *Signal Properties* dialog and make the following edits. Enter **!SIG3** into the **Boolean Equation** edit box (it references itself in the Boolean Equation).

3. Choose **CLK0** from the Clock drop down list box.

4. Check the **Simulate** radio button. Notice that SIG3 signal is completely gray but that the simulation status bar says **Simulation Good**. This is because SIG3's Boolean equation references itself but it does not provide the simulator with a known start state.

5. Press the **Advanced Register** Advanced Register button to open the *Advanced Register and Latch Controls* dialog. This dialog sets the propagation times, setup/hold constraints, clock enable, and set/clear options for this specific register. Note: the global defaults can be set using the **Options > Simulation Preferences** menu.

6. Make sure the **Active Low** and **Asynchronous** check boxes in the "Set and Clear" section are checked. Press **OK** to close the dialog.



Advanced Register and Latch Controls

Set and Clear
✓ Active Low
✓ Asynchronous

Clock Enable
✓ Active Low

Clock to Out: 0 ns
Setup: 0 ns
Hold: 0 ns

7. Choose **SIG0** in the **Set** drop down list box of the *Signal Properties* dialog.

8. Press the **Simulate Once** button. Notice that SIG3 now has a simulated waveform. Experiment with SIG0 to see how the active low set line affects the operation of the flip-flop.

The **Clock to Out**, **Setup**, and **Hold** edit boxes in the *Advanced Register* dialog accept time values for various timing constraints on the register and latch circuit. For more information on Register and Latch timing read *Chapter 12*.

1e) Multi-bit Equations

WaveFormer Pro, TestBencher Pro and VeriLogger Pro can automatically generate multi-bit equations for the register, latch and combinational logic circuits. To convert a register or latch circuit into a multi-bit signal, change the MSB of the input signal and the MSB of the register or latch. If the sizes of the signals do not match, WaveFormer maps as many LSB's as it can.

First, set up the diagram to experiment with multi-bit equations:

1. **Delete** the **SIG3** signal.
2. Create a copy of SIG2. Click on the SIG2 name in the Label window to select it, **Use the Edit > Copy Signals** menu option to copy the signal, then the **Edit > Paste Signals** option to paste the signal. There are now two SIG2s in your diagram. Rename the bottom SIG2 to SIG20. SIG20 should have the exact same waveform as SIG2.

Next, change the output of SIG20 to a multi-bit signal:

1. Double click on the **SIG20** signal name to edit it in the *Signal Properties* dialog.
2. Make sure the **Simulate** radio button is selected.
3. Type **3** into the **MSB** edit box of SIG20. This will make SIG20 a 4-bit signal.
4. Click the **Apply** button. SIG20's waveform is now drawn as a bus with a 4 bit binary display. Only the LSB of SIG20 is working because the input signal SIG1 is a single bit. Compare SIG2 and SIG20 and verify that they are the same values.

Change the input signal SIG1 to a multi-bit signal:

1. Double click on the **SIG1** signal name to edit it the Signal Properties dialog.
2. Change the name of **SIG1** to **SIG1[3:0]**. Changing the name using the bracket notation has the same effect as changing the values in the MSB and LSB edit boxes.
3. Click the **Apply** button to accept the change. Now all four bits of SIG20 should be toggling 1111 and 0000.

If you want to further experiment with multi-bit signals, change SIG1's graphical segments to Valid regions instead of Highs and Lows. Then double click on a valid region to open the Edit Bus State dialog box. Type different 4-bit values, like 1010 or 0011, into the Virtual edit box and watch how it affects the output of SIG20.

Next, setup the diagram for the next section:

1. Delete signals **SIG1**, **SIG2**, and **SIG20** by selecting the names and pressing the **Delete** key.
2. Add a new signal called **SIG1**. DO NOT draw the waveform, we will simulate it in the next section.
3. The timing diagram should consist of one clock (CLK0), and 2 signals (SIG1 and SIG0).

1f) Experiment with Behavioral HDL Code

In addition to the simulation of Boolean and registered logic circuits, SynaptiCAD products can simulate behavioral HDL code. To enter behavioral code for a signal, click on the HDL code button in the *Signal Properties* dialog and type code directly into the edit box.

WaveFormer Pro, VeriLogger Pro and TestBencher Pro also provide a template feature that lets you auto generate the register and latch models used by the Logic Wizard. In this section we will use a register template as a starting point to build a circuit that asynchronously counts the number of edges that occur on SIG1 and synchronously presents the total number of edges on the negative edge of the clock. To model this circuit:

1. Double click on the **SIG1** signal name to edit it in the *Signal Properties* dialog.
2. Choose **CLK0** from the **Clock** drop down list box.
3. Choose **neg** from the **Edge/Level** drop down list box.
4. Type **3** into the **MSB** edit box. DO NOT enter anything into the Boolean Equation edit box.
5. Click the HDL code button to view the resulting template code:

```
wire  [3:0] SIG1_wf7  = PLACEHOLDER ;
registerN_Asyn #(4,0,0,0,0) registerN_Asyn_SIG1(SIG1, CLK0, SIG1_wf7,1'b0,1'b1,1'b1);
```

Note: the internal wire name SIG1_wf*** will vary depending on how many signals you have simulated.

The auto generated variable **PLACEHOLDER** is undefined and will not simulate. If a Boolean equation was defined for the circuit, it would replace the PLACEHOLDER variable. The **registerN_Asyn** line instantiates (defines an instance of) a 4 bit negative-edge-triggered register of the type used by the logic wizard. This register takes PLACEHOLDER as an input and outputs a synchronized version on SIG1.

6. We will use the PLACEHOLDER variable to store the edge count. Edit the behavioral code so that it looks like this (add the bold lines):

```
reg [3:0] PLACEHOLDER;
initial PLACEHOLDER = 0;
always @(SIG0)
 PLACEHOLDER = PLACEHOLDER + 1;
wire  [3:0] SIG1_wf7  = PLACEHOLDER ;
registerN_Asyn #(4,0,0,0,0) registerN_Asyn_SIG1(SIG1, CLK0, SIG1_wf7,1'b0,1'b1,1'b1);
```

7. Check the **Simulate** radio button. Verify that SIG1 is counting the edges of SIG0. The new edge count is presented on each negative edge of CLK0.

The code that you just entered is behavioral Verilog code. The first line defines PLACEHOLDER as a 4-bit register. PLACEHOLDER needs to be defined as a register rather than a wire in this case because it must "remember" its value. Verilog wires don't remember their values so they must be constantly driven to retain their value. The 2nd line initializes the value of PLACEHOLDER to 0 when the simulator first runs. The 3rd and 4th lines contain an always block (note for VHDL users: these work like VHDL process blocks). Whenever SIG0 changes state, the always block will execute, incrementing PLACEHOLDER. The last 2 lines consist of the automatically generated template code that instantiates the synchronizing register.

Tip: More information on the HDL simulator can be found in *Chapter 12* in the manual and the on-line help. Also, the *Advanced Modeling and Simulation* turoail demonstrates how to model a complex circuit using external models, behavioral HDL code, and incremental simulation techniques. The HDL Simulation features are different from the VHDL and Verilog test bench generation features which are covered in the Advanced HDL Simulation, Verilogger Pro, and TestBencher Pro tutorials.

2) Advanced Waveform Generation

In this section you will learn techniques to quickly generate signals from temporal equations, rearrange signals, and work with 2 different types of buses. These features are contained in all SynaptiCAD products.

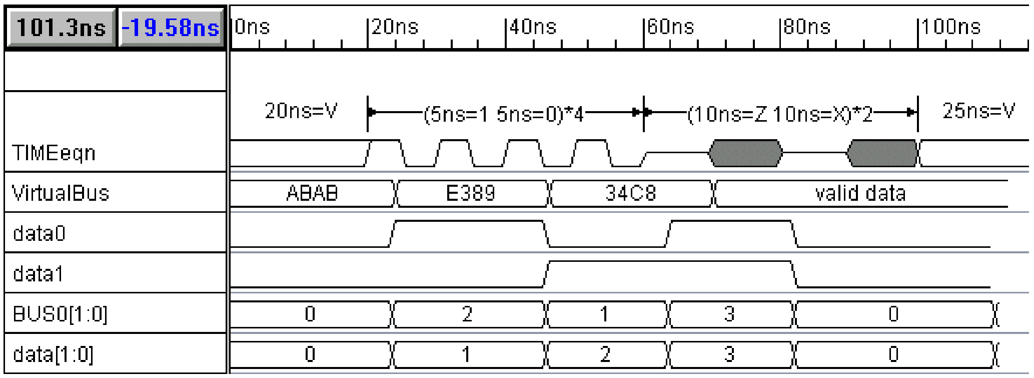


Figure 2.1 Timing diagram used for the Advanced Waveform Generation tutorial. Demonstrates busses, virtual busses, and signal generation using temporal equations.

Let's start a new timing diagram for this section:

1. Select the **File > New** menu option to start a new diagram. You can either save or not save the current diagram.

2a) Generate Waveforms From Temporal Equations

Generating waveforms from temporal equations provides a quick way to generate signals that have a known pattern that is more complicated than a periodic clock.

Temporal equations are entered in the *Signal Properties* dialog using the edit box to the right of the **Wfm Eqn** button. The edit box contains the default equation: **8ns=Z (5=1 5=0)*5 9=H 9=L 5=V 5=X**

The default equation draws a waveform that uses all of the available waveform states. If you start by editing the default equation you do not have to memorize the syntax of these equations.

The syntax consists of space-separated, time-value pairs. The values are 0, 1, Z, V, H, L, and X which represent the graphical states of the waveforms. For example, the **8ns=Z** part of the default equation draws an **8 ns tristate** segment.

To repeat a sequence of states, enclose a list of time-value pairs in parentheses and use the multiply symbol '*' followed by the number of times the list is to be repeated. For example, **(5=1 5=0)*5** draws five copies of a 5ns strong high segment followed by a 5ns strong low segment.

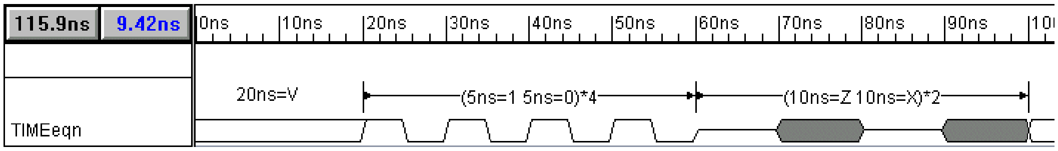
To experiment with temporal equations:

1. Click the **Add Signal** button to add a signal, and change its name to **TIMEeqn** using the *Signal Properties* dialog. (If the dialog is closed then double click on the name to open it.)
2. In the waveform equation edit box, (to the right of the **Wfm Eqn** button) enter the equation:

   20ns=V (5ns=1 5ns=0)*4 (10ns=Z 10ns=X)*2 25ns=V

3. Click on the **Wfm Eqn** button to apply the equation.

The 20ns=V in the equation means that the signal will be initially valid state for 20ns. Next the (5ns=1 5ns=0), will cause the signal to be a strong high for 5ns then a strong low for another 5ns. The *4 will cause the sequence inside the parentheses to be repeated 4 times. The rest of the equation is interpreted in the same way.



The text in the diagram above is made with a combination of text objects and setup parameters with custom labels. It is used to illustrate the different components of a temporal equation. This is just a quick demonstration of the documentation abilities of the program. For more information on documentation read *Chapter 8*. You do not have to add the text for this tutorial.

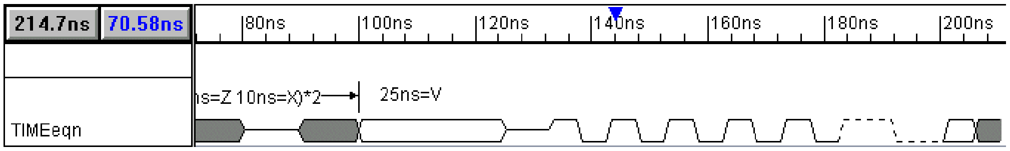
Waveform equations are stored in the waveform equation drop down box located next to the **Wfm Eqn** button.

The equations can be used to create new signals or concatenated to the end of an existing signal.

Adding equations to existing signals:

1. In the *Signal Properties* dialog, click on the down arrow of the equation drop-down box to display the previous equations.
2. Select the default equation **8ns=Z (5=1 5=0)*5 9=H 9=L 5=V 5=X**. You may have to scroll down to find it.

3. Click the **Wfm Eqn** button. Notice that the waveform was added to the end of the TIMEeqn signal.



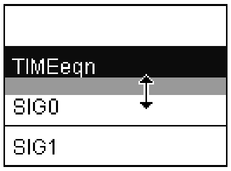
Temporal Equations and a related feature called State Label Equations provide a quick method of generating and then labeling signals that represent Counter and Shifter circuits. See *Chapter 11: Waveform Equation Generation* for more information on these features.

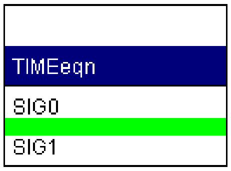
2b) Moving and Reordering Signals

All signals are moved by dragging and dropping the signal's name. When several signals are highlighted and moved as a group they will reorder themselves according to the order in which they are selected. This ability to quickly reorder signals by the order of selection will help you deal with the large numbers of member signals of buses.

Moving a single signal:

1. Click on the **Add Signal** button twice to create 2 new signals.
2. Select the signal **TIMEeqn** by left clicking on the signal name. A selected signal is highlighted.
3. Put the mouse cursor near the very top of TIMEeqn. When the mouse cursor changes from a normal arrow to an up/down arrow, click down and hold the left mouse button. A green bar will appear.
4. **Drag** the green bar until it is in between SIG0 and SIG1.
5. **Drop** the green bar by releasing the left mouse button. Notice that the timing diagram has redrawn itself.
6. Try dropping TIMEeqn at the very bottom and the very top of the diagram.





Moving and reordering multiple signals:

1. Select **SIG0**, then select **SIG1** by left clicking on the signal names in that order.
2. **Move** the signals to the bottom of the diagram. Notice that SIG0 is above SIG1 because that is the order in which they were selected.
3. Select **SIG1** and then select **SIG0**.
4. **Move** the signals to the top of the diagram. Notice that SIG1 is above SIG0, because the signals were selected in that order. This is a quick way to reorder a large group of signals.

5. **Delete** SIG0 and SIG1. They are not used in the next section.

2c) Group Buses and Virtual Buses

There are two kinds of buses supported by the timing diagram editor: group buses and virtual buses.

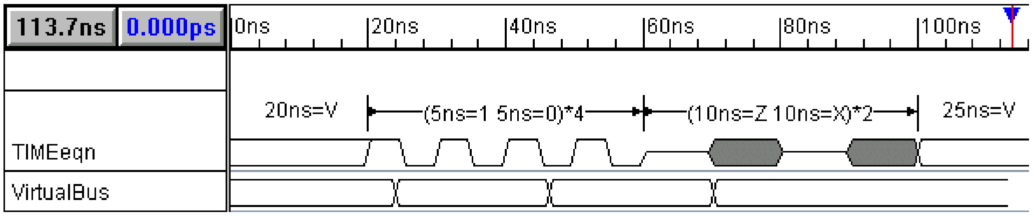
- **Group buses** are composite signals whose transitions and state values are determined by their member signals. Instead of individually editing related signals (like the address lines of a part), a group bus can compress all the signals' data into one compact signal. The individual member signals can be uncoupled, or displayed along with the bus. Buses are added using the **Add Bus** button.

- **Virtual buses** are normal signals that use extended state information to represent bus values. Virtual buses are added using the **Add Signal** button. The state information is added using the HEX state button and the ExState edit box. A virtual bus does not have member signals.

Creating Virtual Buses:

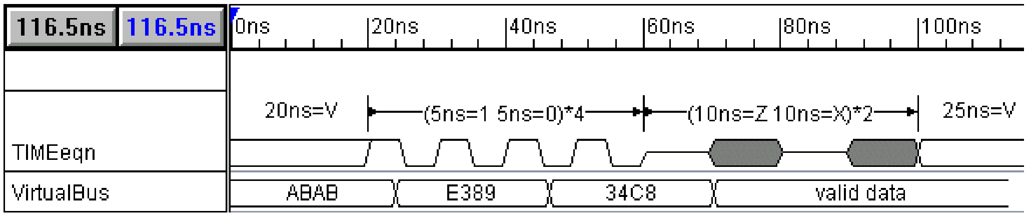
Virtual Buses are the recommended way to display and work with bus information. Virtual Buses are also supported by the VHDL and Verilog stimulus and test bench generation features. If timing parameters are attached to a bus then virtual buses will increase computational performance for timing diagrams that use large buses (32 bits or more). To create a virtual bus:

1. Click the **Add Signal** button to add a new signal and name it **VirtualBus**.
2. Click twice on the **VAL** state button so that the Valid state button stays active (state buttons will not toggle). The Valid button should be red and have a red T at the top of the button.
3. **Draw** four consecutive valid segments similar to the VirtualBus signal in the figure below.



4. Double left click on the **first** segment in VirtualBus to open the *Edit Bus State* dialog box.
5. Enter the data into the Virtual field and use either the **Next** and **Previous** buttons or the key combinations **Alt-N** and **Alt-P** to move between the different segments. Any string of characters and numbers can be displayed in the bus. We used the following data: **ABAB**, **E389**, **34C8**, **valid data**.
6. Click **OK** to close the *Edit Bus State* dialog when all the segments have been edited.
7. Click the **ZOOM OUT** button several times to demonstrate how the extended state data automatically hides itself when its segment becomes too small to display the text.

8. Click the **ZOOM IN** button several times and return the diagram to its original zoom level.



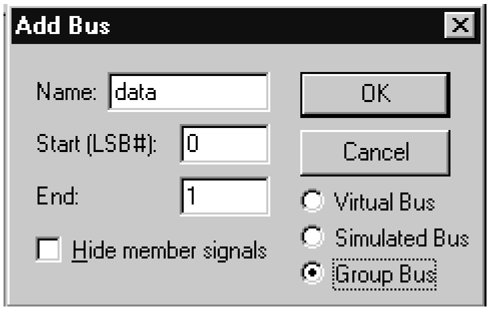
When exporting to VHDL or Verilog, the Virtual State information contained in a valid supersedes the graphical state of a segment. This allows you to export the state values of signals with types that have no graphical representation (integers for example).

Creating Group Busses:

Use group buses only when you need to get access to an individual bus signal at some point in your design or if you need to compress several signals that already exist. Group buses are useful for analysis of data that is imported from simulators or test equipment. Before a group bus can be created, its member signals must either be specified by selecting the signal names or new signals need to be created. We will use both methods in this tutorial.

To create a group bus and its member signals:

1. Make sure that no signal names are selected (clear selected signals by clicking in the Diagram window).
2. Left click on the **Add Bus** button. This will open the *Add Bus* dialog box.
3. Type **data** into the **Name** box. The member signals will be named the same name as the bus, plus their signal number.
4. Enter **0** into the **Start (LSB#):** edit box. This is the least significant bit of the bus.
5. Enter **1** into the **End (MSB#):** edit box. This is the most significant bit of the bus.
6. Verify that the **Hide member signals** checkbox is **NOT** checked. We want to be able to see the member signals in this demonstration.


Add Bus

Name: data

Start (LSB#): 0

End: 1

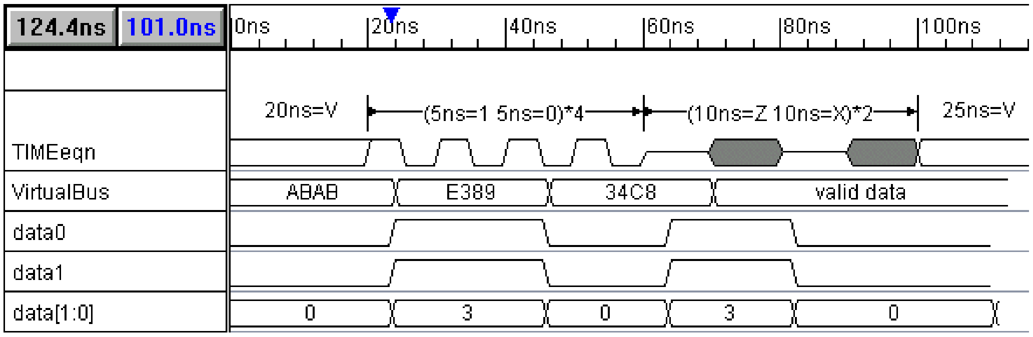
☐ Hide member signals

Virtual Bus [ ] Simulated Bus [ ] [✓ Group Bus]

7. Make sure the **Group Bus** radio button is selected. The radio buttons provide an easy method for creating group or virtual buses.

8. Click the **OK** button to create the bus. There should be 3 signals generated: data (the bus), and data0 and data1 (the bus member signals). If the member signals are not shown, use the **View > Show Hidden Signals** menu option to unhide them.

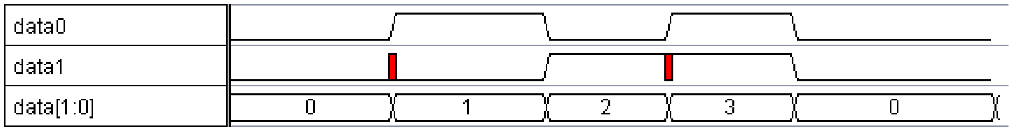
9. Next, **draw** 5 high and low segments on **data** (the bus signal) and notice that the member signals are automatically drawn.

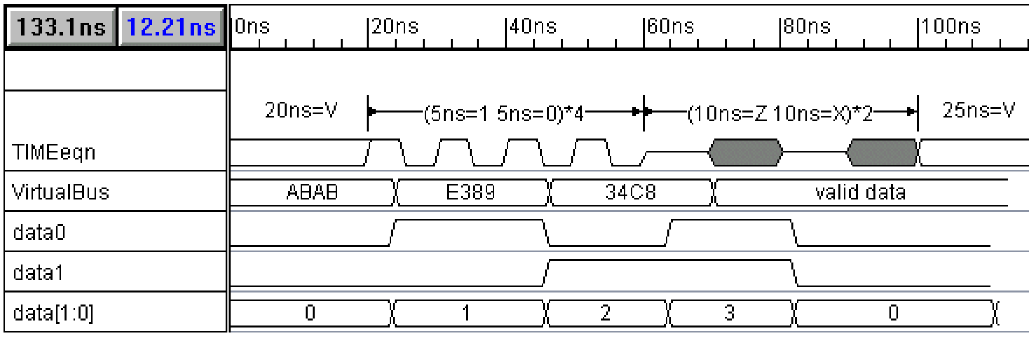


10. Double left click on the first segment of **data** to open the *Edit Bus State* dialog.

11. Type the value **0** into the **Hex** edit box.

12. Use the **Alt-N** key combination to move to the next segment. Enter the values: **1,2,3,0** into the remaining four segments. Notice that the member signals have redrawn properly (except the red transition markers which we will fix later). The red transition markers preserve all the edge information of the member signals during a bus editing session. Click **OK** to close the dialog.



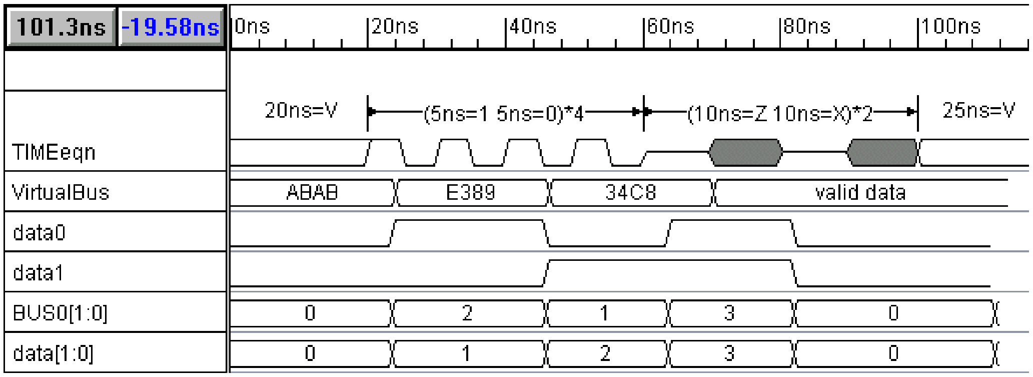


13. Select the **Edit > Clear Red Events** menu option to remove the edge place holders on the member signals.

To create a group bus from existing signals:

Select the signal names to be grouped, in order from LSB to MSB, then click the Add Bus button. In the next example we will create a second group bus whose member signals are reversed from the data bus.

1. **Select** signal **data1** by left clicking on the name. This will be the LSB of the new bus.
2. **Select** signal **data0** by left clicking on the name. This will be the MSB of the new bus.
3. Left mouse click on the **Add Bus** button. Notice that a new bus, **BUS0**, is added to the diagram and that the *New Bus* dialog did not open up because the bus was automatically created from the selected signals. Also note that **BUS0** and **data** have different MSBs and LSBs.



Group buses have many features that are covered in *Chapter 3* of the manual and the on-line help. Before you use group buses extensively, you should read this chapter and become familiar with the **align**, **bind**, and **expand** features.

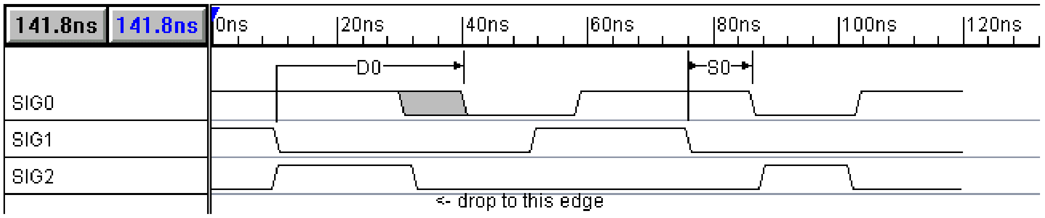
3) Advanced Parameter Techniques

In this section you will learn to control what information a parameter displays using the attribute and custom display strings at an individual and global level. You will learn how to change the signal transition a parameter is attached to, and how to control the vertical placement of a parameter. These techniques will allow you to control exactly what your timing diagrams look like and what information is displayed. These techniques can be used in all four products: Timing Diagrammer Pro, WaveFormer Pro, VeriLogger Pro, and TestBencher Pro.

Setup a timing diagram for this section:

1. Open the file **tutadv11.tim**.

2. Select the **File > Save As** menu option, and save this file as **Mytut2.tim**.

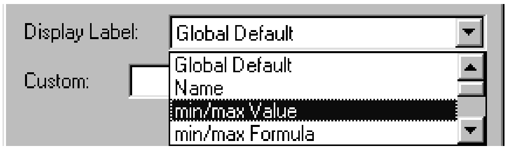


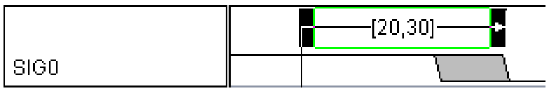
3a) Individual Parameter Display

By default, the text displayed by parameters in the Diagram window is controlled by the **Options > Drawing Preferences** dialog box. However, you can make a parameter display a specific attribute or a custom display string by using the *Parameter Properties* dialog box. The name of the *Parameter Properties* dialog reflects the type of parameter that is currently being edited. For simplicity we will call the dialog *Parameter Properties*, even though the name at the top may say *Delay Properties* or *Setup Properties*.

Editing individual parameter labels:

1. Double left click on the delay label **D0** in the Diagram window. This opens the *Parameter Properties* dialog box.
2. Use the **Display Label** drop-down list box to select the **min/max Value** display. Notice that the label for the parameter now displays [20,30] - the min/max value - instead of the name D0.





3. Click the **Next** button to display the setup **S0** in the *Parameter Properties* dialog.
4. Use the **Display Label** drop-down list box to select the **min/max Margin** display. The label now shows the margin for the setup. Margin is the amount of time available before a setup or hold constraint is violated.
5. Use the **Display Label** drop-down list box to select the **Distance** display. The label now shows the minimum and maximum distances between the transitions.

3b) Customizing a Parameter's Display

A parameter label can be made to show more than one attribute or to show a custom string of characters and attributes using the **Custom string** in the *Parameter Properties* dialog box. In a custom string, certain character sequences are interpreted as attribute control codes, and when such a sequence is found it is replaced with that parameter's attribute. Attribute control codes start with a **%** character followed by one or two letters. The control codes are:

name = %n

min value = %mv and max value = % Mv

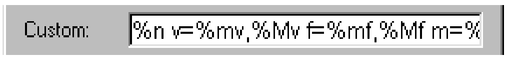
min formula = %mf and max formula = %Mf

min margin = %mm and max margin = % Mm

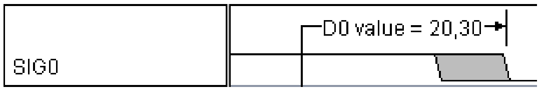
min distance = %md and max distance = %Md

comment = %c

The default custom character sequence contains all the control codes, so it is not necessary to memorize them.


Custom: %n v=%mv,%Mv f=%mf,%Mf m=%

1. Click the **Previous** button in the *Parameter Properties* dialog to display the parameter D0.
2. Use the **Display Label** drop-down list box to select the **Custom** display.
3. Make sure the default custom string is: **%n v= %mv,%Mv f=%mf,%Mf m=%mm, %Mm d=%md,%Md %c**
4. Notice that the parameter label now displays all the attributes available. The default custom string is a little messy to look at, however it contains all of the control codes so you don't have to remember them. When you want to make a custom label just edit the default string.
5. Next, edit the parameter label to display only the parameter's name and min and max values. Edit the contents of the custom string so that the string reads: **%n value = %mv,%Mv**
6. Click the **Apply** button. The label now shows:



Notice that only the display control codes were replaced by attributes, while all other characters like the comma and the "value =" were displayed exactly as typed.

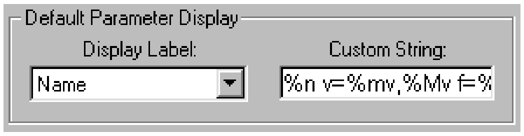
3c) Global Parameter Display

Next, return the parameter labels to their original state:

1. Use the **Display Label** drop-down list box to choose the **Global Default** display for both **D0** and **S0**.
2. Click **OK** to exit the dialog box.

A simple way to have all new parameters display the same attributes is to change the default parameter label. To change the default parameter label:

1. Select the **Options > Drawing Preferences** menu option. This opens the *Drawing Preferences* dialog box. In the bottom half of the *Drawing Preferences* dialog box is the display label drop down combo box which controls the global display for all parameters. "Name" is the default value.



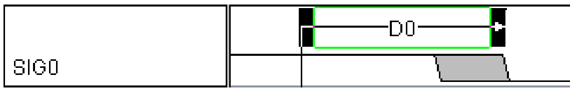
Display Label: Name
Custom String: %n v=%mv,%Mv f=%

2. Left-click on the **Display Label** drop down box to view the list of displayable attributes.
3. Click on the **min/max Value** entry (you may have to scroll down to find it). This changes the global default value from **Name** to **min/max value**.
4. Click **OK** to close the dialog box. Now both the delay and setup labels show the min and max values.
5. Experiment by changing the various values and radio buttons in the **Options > Drawing Preferences** and viewing the effects on the Diagram window.
6. When you are done, left click on the **Defaults** button in the *Drawing Preferences* dialog box to return all options to their default values, and click **OK** to exit.

3d) Drag and Drop Parameter End Points

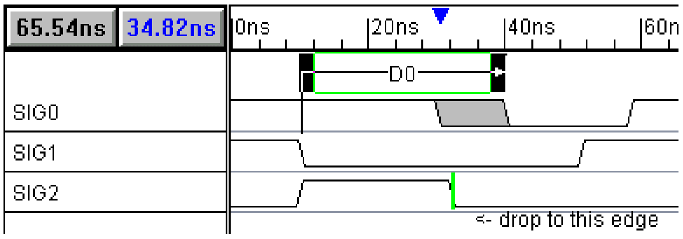
When a parameter is created it is attached to two signal transitions. These signal transitions can be changed by dragging and dropping one of the parameter end-points to a new signal transition. To demonstrate dragging and dropping a parameter's end-point:

1. Left click on the delay parameter D0 to select it. A selected parameter is surrounded by a rectangle with a solid handle box on either end.

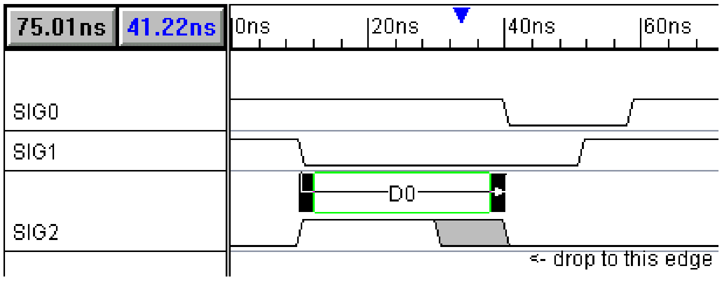


2. Place the mouse over the solid handle box on the right side of the selection rectangle.

3. **Left click down** and **drag** the mouse to the edge indicated on SIG2 so that it is highlighted. If the entire parameter is changing its vertical position then you clicked on the middle of the parameter instead of a handle box.



4. Release the mouse button. Now D0 ends on this transition.



Try experimenting with dragging and dropping the parameter end-points of both the setup and delay. Remember, the delay forces its signal transitions to be a specific distance apart, so when you drop to an edge the timing diagram rearranges itself.

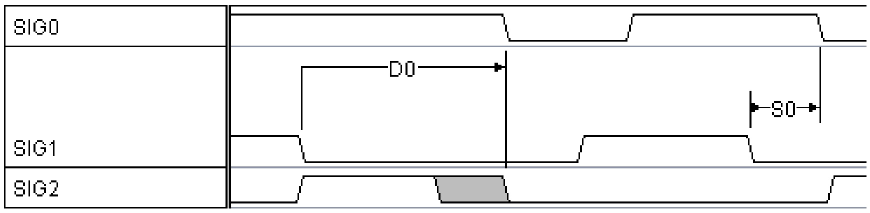
3e) Adjusting the Placement of a Parameter

Normally, the vertical placement for parameters on the screen is set automatically. However, you can also place parameters at a specific height.

To change the placement of a parameter:

1. **Left click** and hold on the center of the delay parameter, **D0**, and **drag** it up to a new vertical position closer to the top of the screen.
2. **Release** the mouse button to place the parameter. Notice that the parameter will stay at that position, even if you add a new parameter to the diagram.
3. **Left click** and hold on the center of the setup parameter, **S0**, and **drag** it down to SIG1's level.

4. Release the mouse button. Notice that the parameter now lies between SIG1 and SIG0.



After you move a parameter, it is considered user placed and it will not be moved from that position unless you choose to move it. Any new parameters will arrange themselves around user placed signals. To return vertical placement control to the program:

1. Open **D0**'s *Parameter Properties* dialog box by double left clicking on the parameter.
2. **Uncheck** the **User Placed** check box. ☐ User Placed
3. Click the **OK** to close the dialog box.
4. Repeat the above instructions for the setup parameter.

Congratulations! You have completed the *Simulation, WaveForm Generation, and Parameters* tutorial.

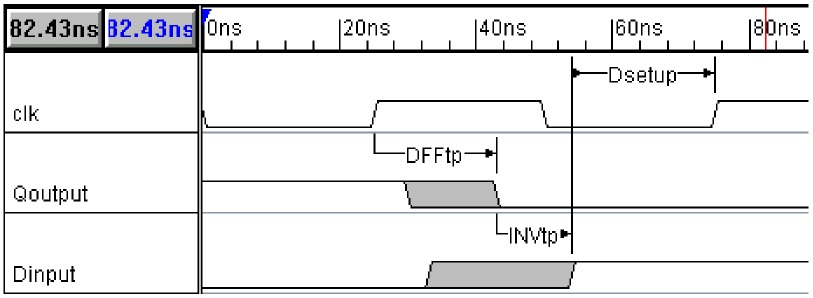
Parameter Libraries

This tutorial explains how to use the library functions of WaveFormer Pro, TestBencher Pro, Timing Diagrammer Pro and VeriLogger Pro. It starts up where the basic tutorial ends. If you do not want to go through the first tutorial, you can find a completed diagram of the first tutorial under the filename tutorial.tim (can be loaded directly by clicking on the tutorial icon). A completed diagram of this tutorial is also on your disk under the filename tutlib.tim if you wish to check against your diagram at the end of this tutorial.

Getting Started

First, configure your program, and load the file tutorial.tim.

1. Minimize the Report window (and Project window if applicable). They are not used in this tutorial.
2. Select the **Window > Tile Horizontally** menu option. Both the Diagram and the Parameter windows should be visible during this tutorial. If you are unable to view one of the windows, use the **Window > Parameter** or **Window > Diagram** menu option to open the missing window.
3. Select the **File > Open Timing Diagram** menu option.
4. Select the **tutorial.tim** file from the directory where your product is installed.
5. Click **Open** to load the file.



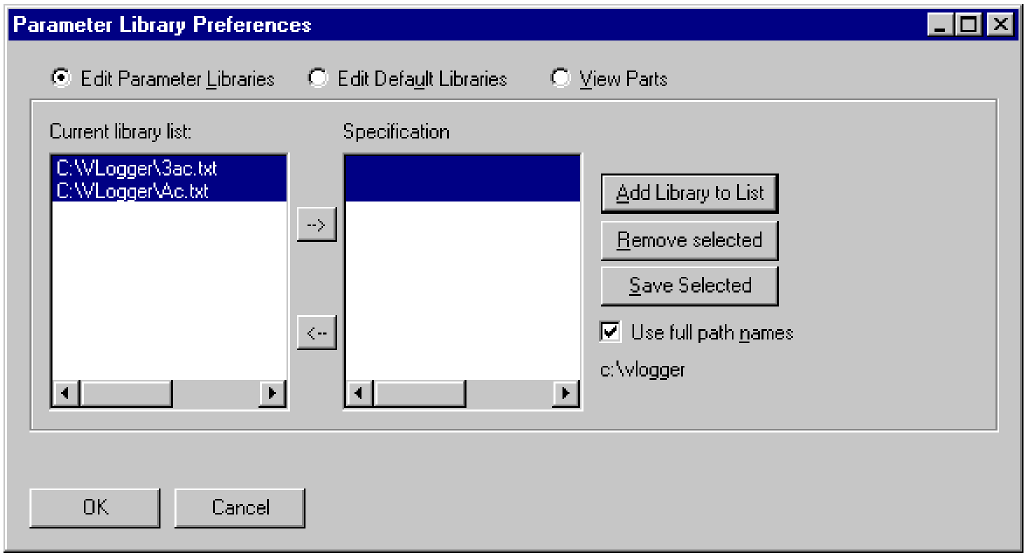
1) Adding Parameter Libraries to the project's "Library Search List"

In order for a project to use a library, it must know the library's name and path location. This information is kept in the project's library search list.

To edit the library search list of the tutorial.tim file:

1. Select the **Libraries > Parameter Library Preferences** menu option. This opens the *Parameter Library Preferences* dialog.

2. Click the **Add Library to List** button Add Library to List to open the *Parameter Library Browse* dialog to search for libraries on your disk.
3. Select the two sample libraries **ac.txt** and **3ac.txt**.
4. Click the **Open** button to add the selected files to your search list and close the *Parameter Library Browse* dialog.



(● Edit Parameter Libraries) ○ Edit Default Libraries ○ View Parts

Current library list:
C:\VLogger\3ac.txt
C:\VLogger\Ac.txt

[✓] Use full path names

c:\vlogger

Now in the *Parameter Library Preferences* dialog, both libraries should be selected in the library search list path section. The next section also uses the *Parameter Library Preferences* dialog so leave it open.

Note: the filenames for the libraries will have their path names attached unless you have unchecked the "Use full path names" checkbox. Generally you will want to leave this option checked as this allows you to use libraries in multiple directories.

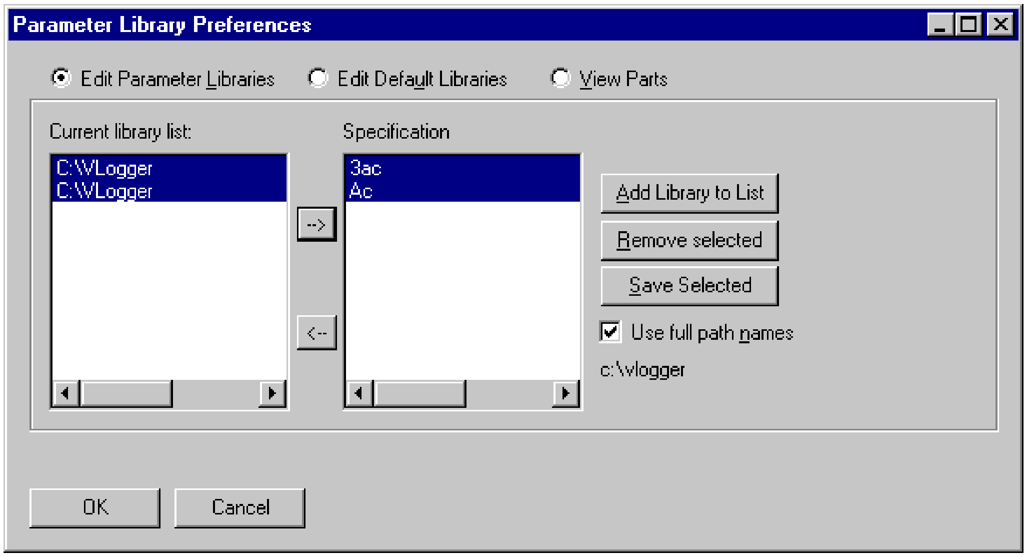
2) Setting Parameter Library Specifications

After adding libraries to the project's search list, you need to define the library specification. Library specifications allow SynaptiCAD products to distinguish between similarly named parts in different parameter libraries. Libraries 3ac.txt and ac.txt contain parameters with the same names. If you do not assign specifications and you referenced these parameter names in your design, the values from the first library in the list would be used.

To assign specifications:

1. Select both the **3act.txt** and the **ac.txt** library files.
2. Click on the arrow button pointing right in the *Parameter Library Preferences* dialog. This will assign specifications to the selected libraries.

Now the specification for ac.txt is "Ac" and the specification for 3ac.txt is "3ac". To eliminate the specification for a library, select it and press the left arrow button.


Parameter Library Preferences

(● Edit Parameter Libraries) ○ Edit Default Libraries ○ View Parts

Current library list:
C:\VLogger
C:\VLogger

Specification
3ac
Ac

[✓] Use full path names
c:\vlogger

3) Startup Parameter Library Configuration

Another useful feature of the Parameter Library Preferences dialog which we will not use in this tutorial is the **Edit Parameter Libraries** and **the Edit Default Libraries** radio buttons. The Edit Parameter Libraries radio button should currently be selected. This allows you to change the parameter library settings for the current project. If you have a set of libraries that you wish to use with all new projects, select the **Edit Default Libraries** radio button and add these libraries to the Startup library search list. Thee libraries will not be added to the current project, but any new project will automatically have these libraries included in their library search list.

Close the Parameter Library Preferences dialog:

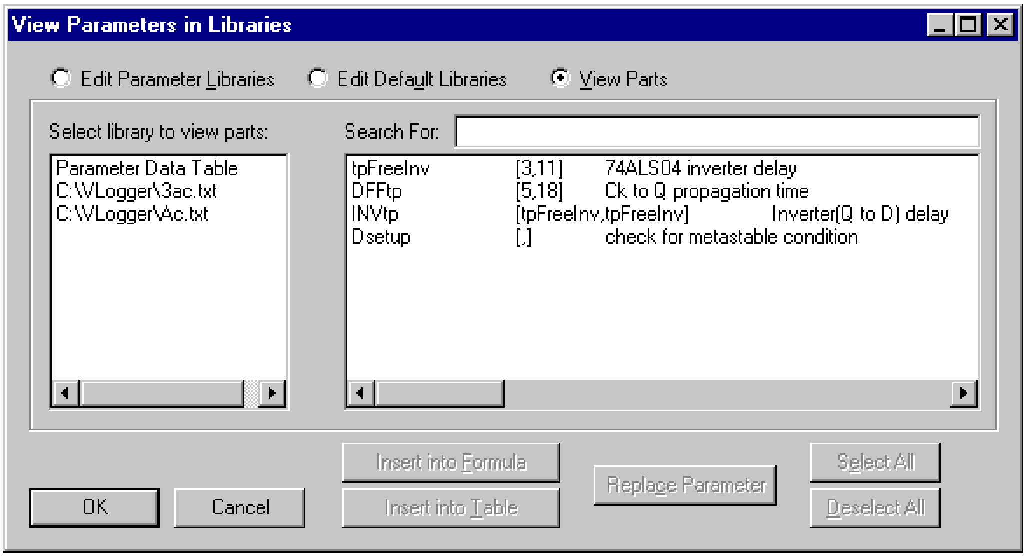
1. Make sure the **Edit Parameter Libraries** radio button is selected.
2. Click the **OK** button to close the *Parameter Library Preferences* dialog.

4) Referencing Parameters in Libraries

Now that we have added the libraries and set the specifications, we want to reference the library parameters in our project.

1. Double click on the **min value** of the **DSetup parameter** (in the parameter window) to edit the value. This opens the *Parameter Properties* dialog box.
2. Delete the value in the **min** edit box, then press the **Library** button to open the *View Parameters in Libraries* dialog.

   - Notice that there are **three** libraries on the library list; the 3ac.txt and Ac.txt that you added, and one called Parameter Data Table. This extra library is a virtual library that lists all the user-added parameters in the project. You can use virtual library parameters in formulas just like regular library parameters.

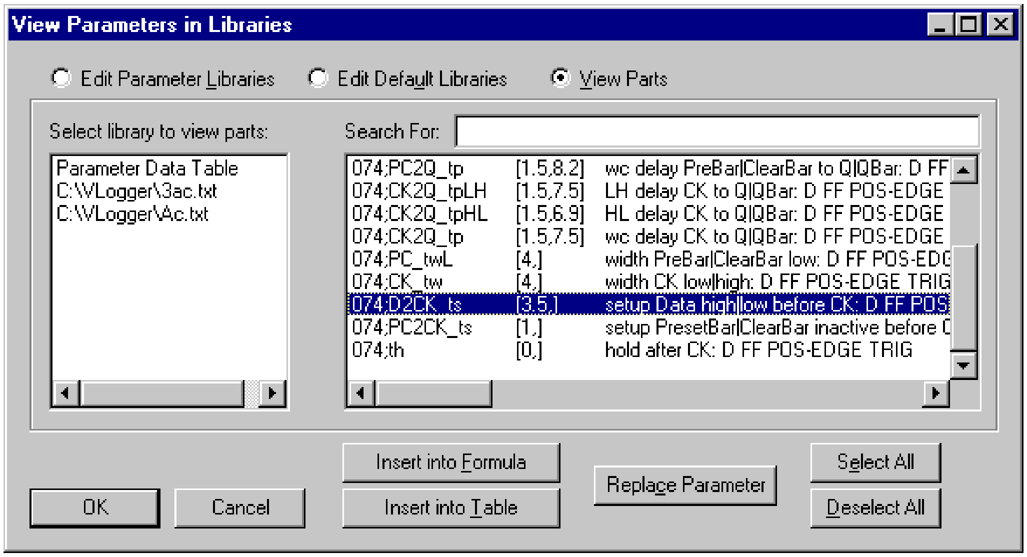


Parameter Data Table
C:\VLogger\3ac.txt
C:\VLogger\Ac.txt

tpFreeInv	[3,11]	74ALS04 inverter delay
DFFtp	[5,18]	Ck to Q propagation time
INVtp	[tpFreeInv,tpFreeInv]	Inverter(Q to D) delay
Dsetup	[,]	check for metastable condition

3. Select the **ac.txt** library from the library list. The parameters in this library are displayed in the Library parts list.

4. Scroll down in the library parts list to find the parameter **074;D2CK_ts**. Left click to select the parameter, and press the **Insert Into Formula** button Insert into Formula.



( ) Edit Parameter Libraries ( ) Edit Default Libraries [✓ View Parts]

Select library to view parts:

Parameter Data Table
C:\VLogger\3ac.txt
C:\VLogger\Ac.txt

Search For:

```
074;PC2Q_tp     [1.5,8.2]  wc delay PreBar|ClearBar to Q|QBar: D FF
074;CK2Q_tpLH   [1.5,7.5]  LH delay CK to Q|QBar: D FF POS-EDGE
074;CK2Q_tpHL   [1.5,6.9]  HL delay CK to Q|QBar: D FF POS-EDGE
074;CK2Q_tp     [1.5,7.5]  wc delay CK to Q|QBar: D FF POS-EDGE
074;PC_twL      [4.]       width PreBar|ClearBar low: D FF POS-EDG
074;CK_tw       [4.]       width CK low|high: D FF POS-EDGE TRIG
074;D2CK_ts     [3.5.]     setup Data high|low before CK: D FF POS
074;PC2CK_ts    [1.]       setup PresetBar|ClearBar inactive before C
074;th          [0.]       hold after CK: D FF POS-EDGE TRIG
```

5. Click **OK** to close the *View Parameters in Libraries* dialog.

The min edit box in the *Parameter Properties* dialog contains the name of the parameter we just inserted (Note: the library specification "Ac" is added to the parameter name, separated name by a colon, i.e. **+Ac:074;D2CK_ts**).

Next, we will edit both the min and max value of the delay **INVtp** at the same time.

1. Double click on the **INVtp** parameter (in the parameter window) to update the *Parameter Properties* dialog display with the values for INVtp.
2. Click on the Library button Library to open the *View Parameters in Libraries* dialog.
3. Select the **Ac.txt** library and insert the parameter **004;tp** into min value. (Select the parameter and press the **Insert Into Formula** button).
4. Choose the **OK** button to close the *View Parameters in Libraries* dialog.
   - Notice that the ac:004;tp parameter was added to the values that were already in the min and max edit boxes.
5. Delete the original values from the min and max edit boxes, leaving only the **ac:004;tp** value.
6. Click the **OK** button to close the *Parameter Properties* dialog.

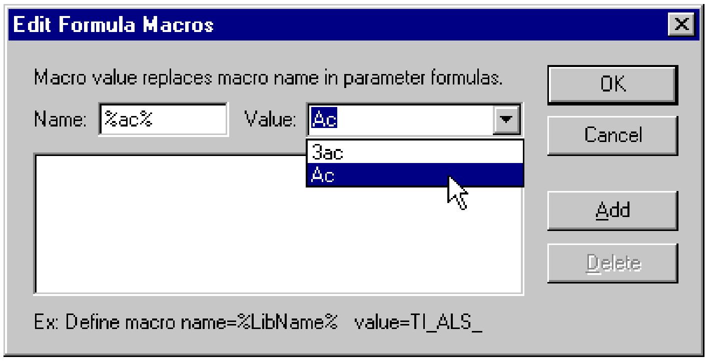
Repeat the above process for the min and max values of DFFtp, inserting **074;CK2Q_tp** from the ac.txt library. Try using the **Search For** edit box in the *View Parameters in Libraries* dialog, instead of scrolling, to find a parameter name.

5) Using Macros to Examine Trade-offs Between Different Libraries

Your diagram is now using values for the AC logic family operating at 5V. If you want to examine the impact of changing your design to 3.3V, you need to change the library specifications of the parameters to "3ac". It can get tedious changing back and forth between different libraries when you have to change the name of each parameter. To avoid this you can create a macro which you use in place of the library specification in your parameter names. Then to change libraries you just need to change the value of the macro.

To create a macro:

1. Select the **Libraries > Macro Substitution List** menu option to open the *Edit Formula Macros* dialog.
2. Enter **%ac%** into the name edit box.
3. Select **Ac** from the **Value drop down box**. The drop down box contains all libraries that have specifications.


Edit Formula Macros

Macro value replaces macro name in parameter formulas.

Name: %ac%   Value: Ac

Ex: Define macro name=%LibName% value=TI_ALS_

4. Click **OK** to add the macro to your macro list.

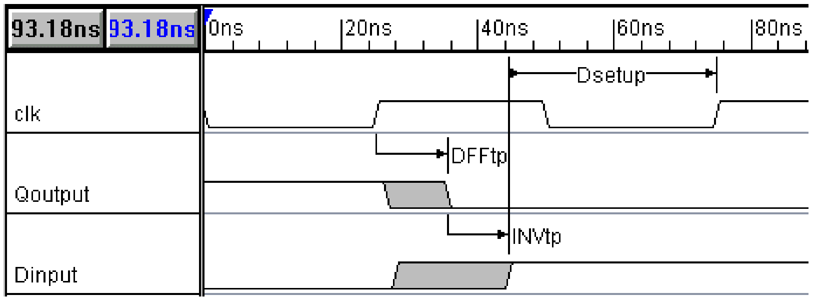
Now edit the five min & max values of your parameters, replacing ac with %ac%. Your design should still be using the 5V AC values. When editing the values, try using the **Next** and **Previous** buttons in the *Parameter Properties* dialog to move between parameters.

To make your design reference the 3V library, change the value of the macro.

1. Select the **Libraries > Macro Substitution List** menu option, to open the *Edit Formula Macros* dialog.

2. Click on the macro **%ac%** in the list box. This places this macro into the Name/Value edit boxes.
3. Use the **Value** drop down box to change the value of the macro to **3ac**. Click **OK** to close the dialog.

Your design should now be using the 3V AC values (the delays should be longer due to the decreased supply voltage). You have now completed the parameter library tutorial.



Note: Macros can also be used to make short or alternative names for library parameters without having to edit the library names.

Advanced HDL Stimulus Generation

This tutorial describes how to generate Verilog and VHDL stimulus files using WaveFormer Pro, VeriLogger Pro and TestBencher Pro. This tutorial is important because it describes exactly how the waveforms of a single timing diagram will be exported. It also covers advanced data types that are used in VHDL generation.

TestBencher Pro customers should also work through the on-line TestBencher Pro tutorials, which cover the sample parameters that generate the self-testing code, and modifying the template files used to generate multi-diagram test benches.

This tutorial covers how the following objects are exported into VHDL and Verilog:

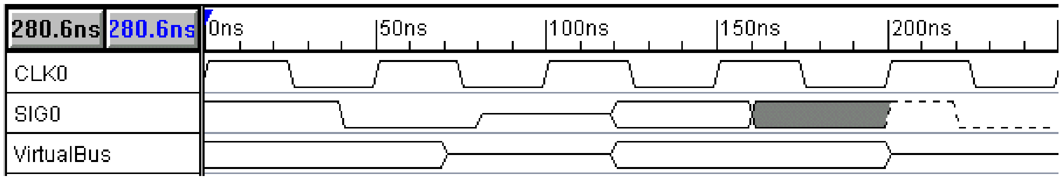
- clocks & signals
- graphical waveform states (high, low, tristate, valid, invalid, weak high, weak low)
- virtual buses with hex, binary, and other data values
- VHDL user-defined types and integer types

1) Getting Started

Get a Full Version License

If you are evaluating WaveFormer Pro, VeriLogger Pro or TestBencher Pro you need to upgrade the evaluation version by obtaining a 30 Day Trial License. This license will turn your evaluation version into a full version for 30 days. The instructions for obtaining a license are located in the evaluation directory under a file named **30day_Permanent License Request.rtf**.

1. Select the **File > Open** menu option and load file **tuthdl1.tim** from the same directory where this tutorial is located.
2. Select the **File > Save As** menu option and save the project as **test.tim** (this will keep the original file intact).



The first signal, *CLK0*, is a clock with a period of 50ns. The second signal, *SIG0*, is a waveform that contains all of the graphical states available in WaveFormer Pro. The third signal, *VirtualBus*, is a waveform drawn with valid and tristate segments.

2) Default Mappings: Hex and Binary Translations

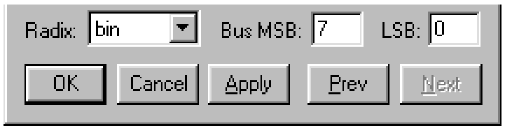
WaveFormer Pro supports a language independent bus format for hexadecimal and binary bus values.

During the translation to Verilog or VHDL, the extended state value of a segment is evaluated to determine if it is a hexadecimal or binary number. If the extended state value begins with a **'b** or **'h** then it is assumed that the number is a binary or hexadecimal number and the number will be translated to the appropriate VHDL or Verilog bus value. If the extended state value does not start with 'b or 'h then the value is written out as it was entered, without any translation.

To demonstrate the hex and binary translations, we will edit the signal VirtualBus so that it will correctly export as an 8-bit bus. We will also use the 'b and 'h values to set the segment values and compare how they export in VHDL and Verilog. Later in the tutorial we will generate the Verilog and VHDL code.

Setting the size of a virtual bus to 8 bits:

1. Double click on the **VirtualBus** signal name to open the *Signal Properties* dialog box.
2. Type **7** into the **Bus MSB** edit box.
3. Type **0** into the **Bus LSB** edit box.


Radix: bin Bus MSB: 7 LSB: 0

4. Click **OK** to close the dialog box.

Setting the values in a virtual bus waveform:

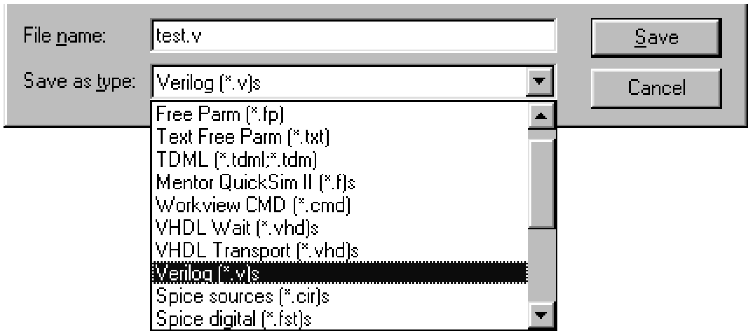
1. **Select** the first waveform segment of **VirtualBus** by clicking on it. A selected segment has a box around it.
2. Click on the **HEX** button at the top of the window to open the *Edit Bus State* dialog box. The *Edit Bus State* dialog box can also be opened by double-clicking on the selected segment.
3. Type **'b11101110** into the **Virtual** edit box. This is an 8-bit binary number.
4. Press **ALT-N** (or press the **Next** button) two times to advance to the next valid segment.
5. Type **'hA** into the **Virtual** edit box. This is the 8-bit hexadecimal number "A" (00001010 in binary). The program automatically left pads missing bits with zeros.
6. Press the **OK** button to close the *Edit Bus State* dialog box.

This behavior of either translating the bus values or straight copying of the information is called the default mapping mode. WaveFormer Pro users can define additional mapping modes by editing the proper stimulus generation file, either **verilog.epl**, **wvhdl.epl**, or **tvhdl.epl** files. These files will be discussed later in the tutorial. To use a new mapping mode, attach an object property to a signal called VhdlMapping or VerilogMapping with a value that is the name of the desired mapping mode routine. TestBencher Pro users will need to edit different files that are discussed in the TestBencher tutorial, but the editing and subsequent use is the same.

3) Generating Verilog Code

Next we will demonstrate how to generate Verilog stimulus vectors from timing diagrams.

1. Select the **Export > Export Timing Diagram As** menu option to open the *Export* dialog.
2. In the **Save as Type** list box in the lower left corner of the dialog, choose the **Verilog (*.v)s script**. This indicates that the timing diagram will be exported to a Verilog code file with a default file extension of ".v". The "s" by convention indicates that a Perl script is being used to do the translation.


File name: test.v
Save as type: Verilog (*.v)s

3. Choose **test.v** as the file name and click the **Save** button to close the dialog. WaveFormer Pro will produce a Verilog file named test.v.
4. The file **test.v** is automatically displayed in the Report window. If you cannot see the Report window, select the **Window > Report Window** menu option to bring the window to the top.

Look at the resulting file by clicking on the **test.v** tab in the Report window. Notice how CLK0 uses a while loop to produce its transitions and how SIG0 uses assignment statements. Also note, values for the VirtualBus assignments have a 8í in front which indicates that VirtualBus is an 8-bit vector. The first segment of VirtualBus has a value of, 8'b11101110, which is the correct Verilog syntax for an 8-bit bus with a binary value of 'b11101110. The next segment has a value of 8'bzzzzzzzz which is the value for an 8-bit tri-stated bus. Next value is 8'b00001010 which is a zero padded translation of the hexadecimal value 'hA.

WaveFormer performs the Verilog stimulus generation using verilog.epl, which is located in the same directory as the main executable. View **verilog.epl** in a report tab by selecting the **Report >**

Open Report Tab menu option. Look at the **%ToState** array. This array determines how the different graphical waveform states are converted to Verilog (SIG0 demonstrates the results of this subroutine). You can modify this Perl script to customize the Verilog export to fit your needs.

When you are done viewing the verilog.epl file, close it with the **Report > Close Report Tab** menu option.

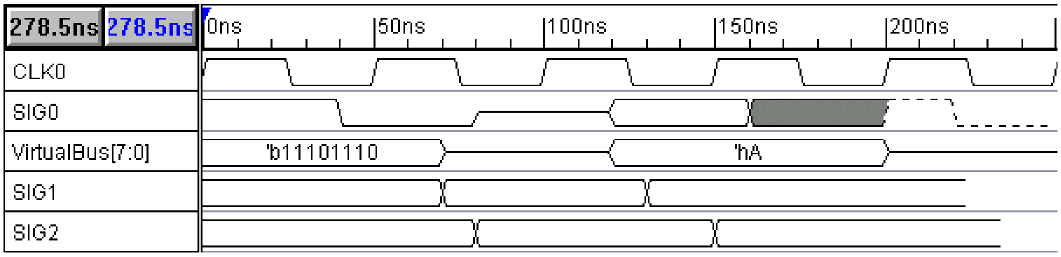
Note: The weak high and weak low signals are not supported by some Verilog simulators. You may wish to map these values to 1 and 0 strong values.

4) VHDL - Advanced Data Types

WaveFormer Pro supports both simple **std_logic** signals and complex user-defined data types for VHDL test benches. By default all signals are assumed to have a type of **std_logic** and a direction of **out** (*CLK0*, *SIG0*, and *VirtualBus* will use the defaults for this tutorial). In this section you will add SIG1 and SIG2 to demonstrate signals with a standard integer type and a user defined type.

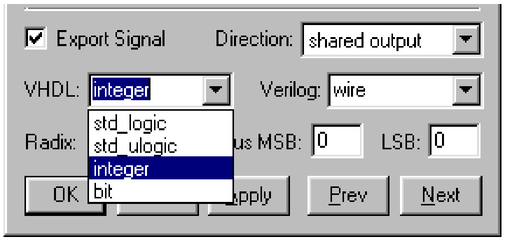
Add SIG1 and SIG2

1. Click on the **Add Signal** button two times to add two signals.
2. Double click on the **VAL** state button. The first click selects Valid as the initial graphical state, and the second click selects Valid as the toggle state (as indicated by the red T). This causes the VAL button to stay selected when you draw waveform segments.
3. Sketch some valid waveforms for SIG1 and SIG2 similar to those in the figure below.



Change the type of SIG1 to integer and the type of SIG2 to MyColor.

1. Double left click on the **SIG1** signal name to open the *Signal Properties* dialog box.
2. Choose **integer** from the **VHDL** drop down list box. You may have to scroll down to find the entry.



3. Select **dec** from the **Radix** drop-down list box to indicate that the values you will be entering into the virtual state are decimal values.
4. Click the **Next** button to move to SIG2. Make sure that you moved to SIG2 and not to the previous *VirtualBus* signal.
5. Enter **MyColor** into the **VHDL** drop down list box. *MyColor* is the name of the user defined type that we will use.
6. Click the **OK** button to close the dialog.

Add Extended State information to the waveforms of SIG1

1. Double click on the **first** waveform segment on *SIG1* to open the *Edit Bus State* dialog box.
2. Type **25** into the **Virtual** edit box and move to the next segment (**Alt-N** or the **Next** button).
3. Enter **integer** values for each segment (we used **25**, **50**, **47**).

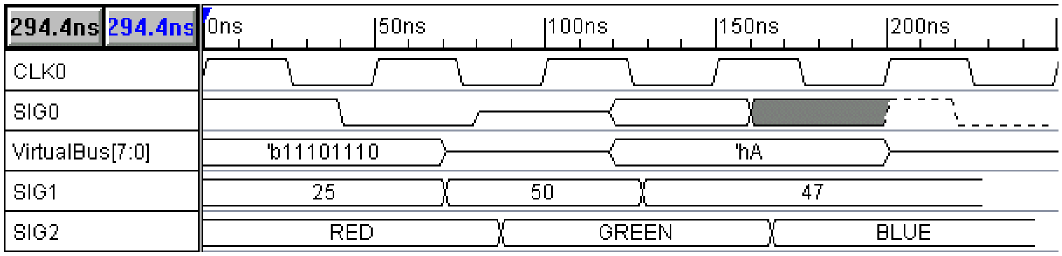
Add Extended State information to the waveforms of SIG2

1. Select the **first** waveform segment on *SIG2*. Notice that the *Edit Bus State* dialog changes to display information about the current selected state.
2. Type **RED** into the Virtual edit box and move to the next segment (**Alt-N** or the **Next** button).
3. Enter color values for each segment (we used **RED**, **GREEN**, **BLUE**).

   For a real VHDL simulation, the color value would have to match the enumerated values defined for MyColor in your original circuit model. For instance, MyColor might be defined as: **enum MyColor ={RED, GREEN, BLUE, BLACK};**

4. Click the **OK** button to close the Edit Bus State dialog and exit the HEX mode.

Your timing diagram should resemble the figure below.



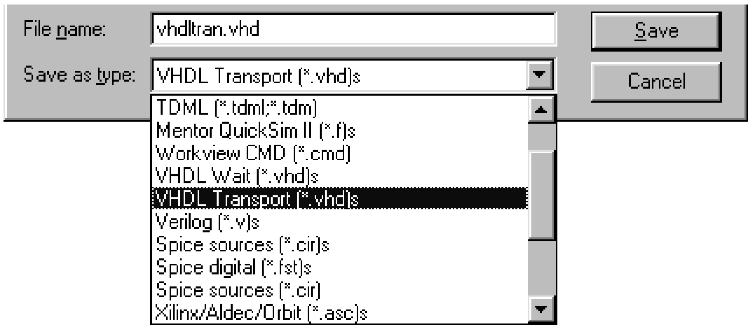
5) Exporting to VHDL

In VHDL there are two ways to write stimulus vectors: using wait statements or using transport statements. Transport-based test benches are smaller and easier to read than wait-based test benches, but wait-based test benches are easier to understand when single-stepping through a simulation to debug it. WaveFormer Pro ships with two different VHDL scripts to create both types of stimulus.

Next, we will export the timing diagram to two different VHDL test benches.

Export to a transport stimulus file:

1. Select the **Export > Export Timing Diagram As** menu option to open the *Save As* dialog.
2. Choose **VHDL transport(*.vhd)s** script using the **Save as Type** list box in the lower left corner of the *Save As* dialog.



File name: vhdltran.vhd

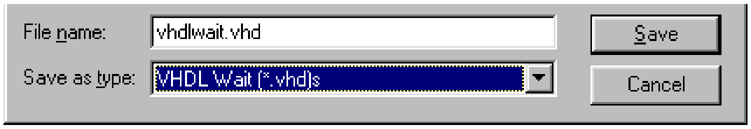
Save as type: VHDL Transport (*.vhd)s

4. Type **VHDLtran.vhd** into the filename edit box.
5. Click **Save** to close the edit box and generate the VHDL transport stimulus file.

Export to wait stimulus file:

1. Select the **Export > Export Timing Diagram As** menu option to open the *Save As* dialog.

2. Select the **VHDL wait(*.vhd)s** script using the **Save as Type** list box in the lower left corner of the *Save As* dialog.
3. Type **VHDLwait.vhd** into the file edit box.



4. Click **Save** to close the edit box and generate the VHDL wait statement stimulus file.

View the file **VHDLtran.vhd** inside the Report window. Notice the entity and architecture structures and the types of all the signals. CLK0 uses a while loop to calculate its value. SIG0 shows how the graphical states are exported. VirtualBus is defined as an 8-bit logic vector. The state values ’b11101110 is translated as 11101110 and ’hA is translated as 00001010. Also note that the tri-stated segments are also exported as 8-bit bus values.

Like VirtualBus, SIG1 and SIG2 use extended state information to define the state values for their waveforms. However since their state values do not begin with **'b** or **'h** the information is used without making any changes to the data.

Congratulations, you have now completed the HDL stimulus generation tutorial. If you have purchased TestBencher Pro, you should read the on-line TestBencher Pro tutorial on generating test benches next (in **tutorial.chm**), otherwise skip to the Advanced Simulation and Modeling tutorial.