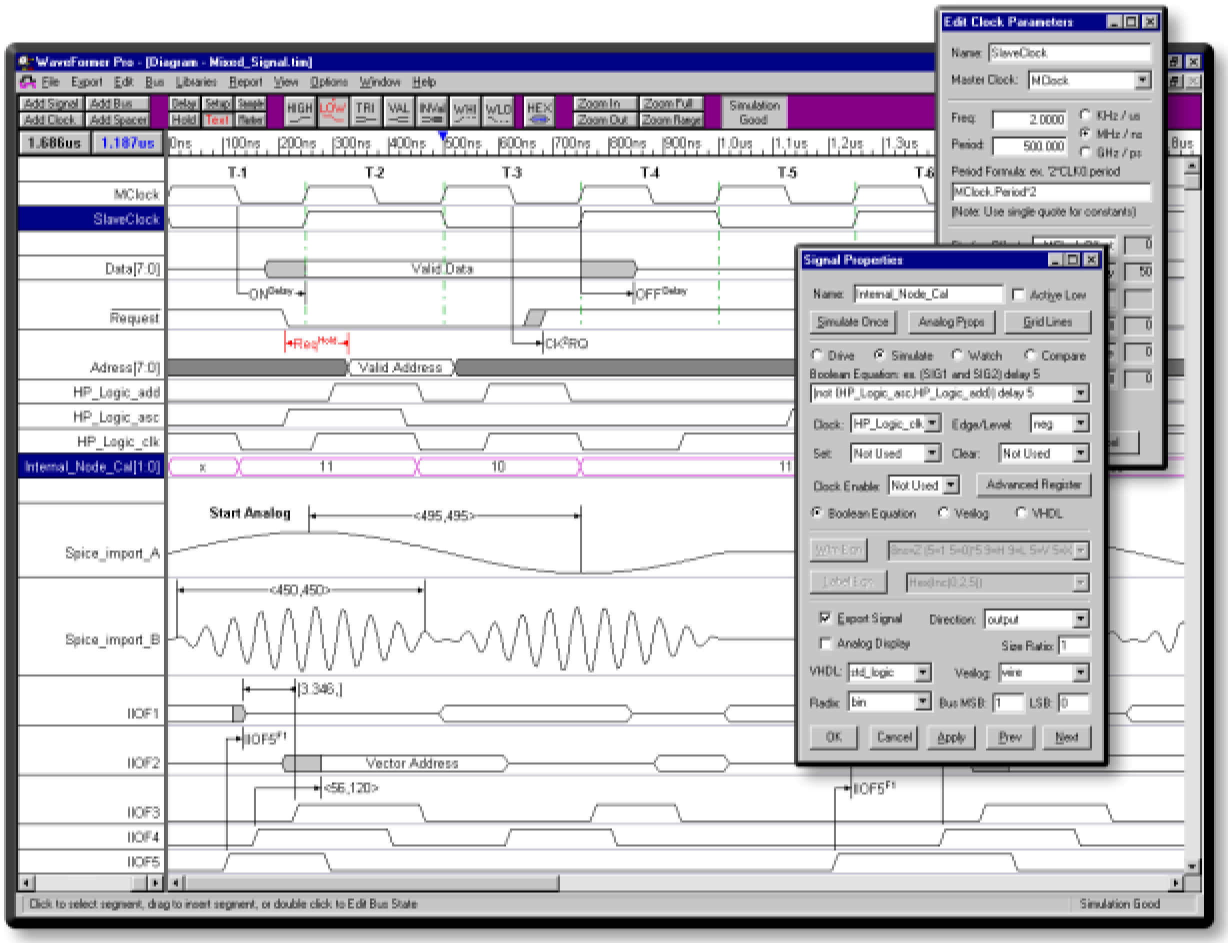
Timing Diagram Editing and Analysis

Copyright © 2011, SynaptiCAD Sales, Inc.


WaveFormer Pro - [Diagram - Mixed_Signal.tim]
File Export Edit Bus Libraries Report View Options Window Help
Add Signal Add Bus Add Clock Add Spacer
Delay Setup Sample Hold Text Marker
HIGH LOW TRI VAL INVal WHI WLO HEX
Zoom In Zoom Full Zoom Out Zoom Range
Simulation Good
1.686us 1.187us
0ns 100ns 200ns 300ns 400ns 500ns 600ns 700ns 800ns 900ns 1.0us 1.1us 1.2us 1.3us
T-1 T-2 T-3 T-4 T-5 T-6
MClock
SlaveClock
Data[7:0]
Valid Data
ON Delay
OFF Delay
Request
Req Hold
Clk2RQ
Adress[7:0]
Valid Address
HP_Logic_add
HP_Logic_asc
HP_Logic_clk
Internal_Node_Cal[1:0]
x 11 10 11
Start Analog
<495,495>
Spice_import_A
<450,450>
Spice_import_B
[3.346,]
IIOF1
IIOF5 F1
IIOF2
Vector Address
<56,120>
IIOF3
IIOF4
IIOF5
Click to select segment, drag to insert segment, or double click to Edit Bus State
Simulation Good
Edit Clock Parameters
Signal Properties
Simulate Once Analog Props Grid Lines
OK Cancel Apply Prev Next
Advanced Register

Edit Clock Parameters

Name: SlaveClock
Master Clock: MClock
Freq: 2.0000
Period: 500.000
KHz / us [✓ MHz / ns] GHz / ps
Period Formula: ex. 2*CLK0.period
MClock.Period*2
(Note: Use single quote for constants)

Signal Properties

Name: Internal_Node_Cal
Active Low
Drive [✓ Simulate] Watch Compare
Boolean Equation: ex. (SIG1 and SIG2) delay 5
(not (HP_Logic_asc,HP_Logic_add)) delay 5
Clock: HP_Logic_clk
Edge/Level: neg
Set: Not Used
Clear: Not Used
Clock Enable: Not Used
[✓ Boolean Equation] Verilog VHDL
[✓ Export Signal]
Direction: output
Analog Display
Size Ratio: 1
VHDL: std_logic
Verilog: wire
Radix: bin
Bus MSB: 1
LSB: 0

Timing Diagram Editing and Analysis

Copyright Copyright © 2011, SynaptiCAD Sales, Inc., version 14

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: January 2011 in (whereever you are located)

Timing Diagram Editing and Analysis

DataSheet Pro, WaveFormer Pro, Timing Diagrammer Pro

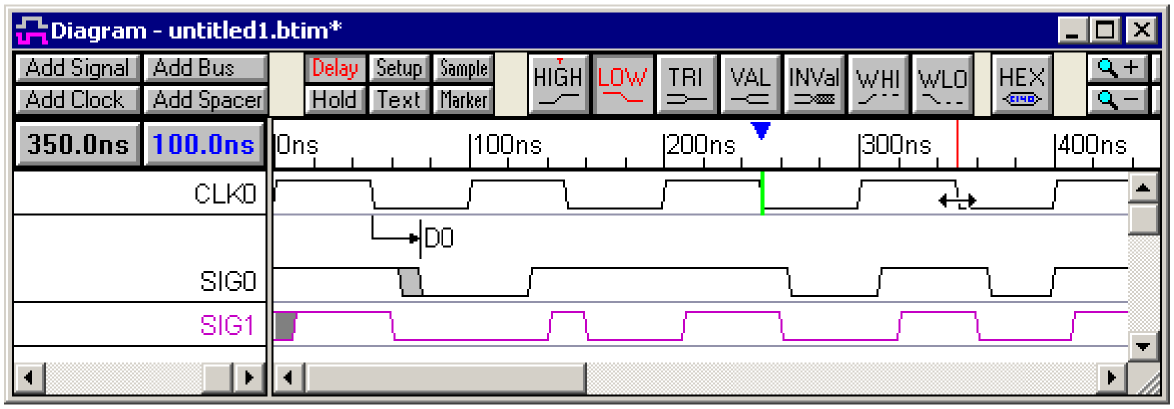
This is the manual for SynaptiCAD's timing diagram editing and analysis features. It is the main manual for WaveFormer Pro, Timing Diagrammer Pro, and DataSheet Pro. It is also a reference manual for the TestBencher Pro, VeriLogger, BugHunter, GigaWave and Transaction Tracker because these products also include some of the drawing and analysis features of the timing diagram editor.

Table of Contents

Copyright © 2011, SynaptiCAD Sales, Inc.

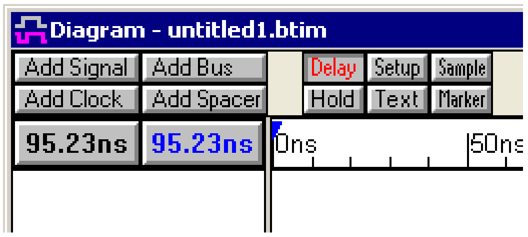
Ultra-Quick Tutorial

This Ultra Quick tutorial shows you how to draw a simple timing diagram and simulate a simple Boolean equation. More comprehensive tutorials are available from the **Help > Tutorials** menu.



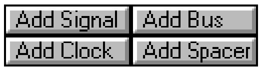
1) Open a new timing diagram file

- Choose **File > New Timing Diagram** menu to open an new timing diagram.

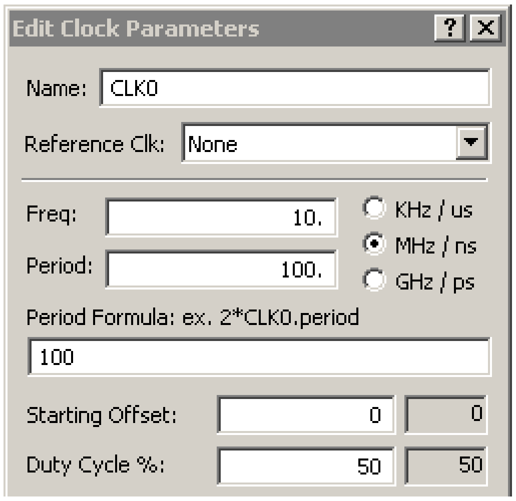


2) Add a clock with a period of 100 ns

- Press the **Add Clock** button to open the *Edit Clock Parameters* dialog.



- The default period is already set to 100 ns, so the clock is created. However, notice that the clock can also be defined using a **reference clock** or a **period formula**. Chapter 2: Clocks 39 covers these options in more detail.
- Press **Ok** to close the dialog.


Edit Clock Parameters

Name: CLK0

Reference Clk: None

Freq: 10.

Period: 100.

KHz / us
[✓ MHz / ns]
GHz / ps

Period Formula: ex. 2*CLK0.period

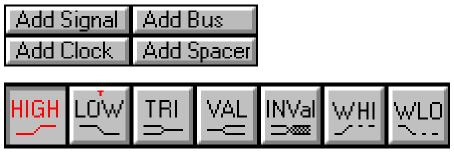
100

Starting Offset: 0 0

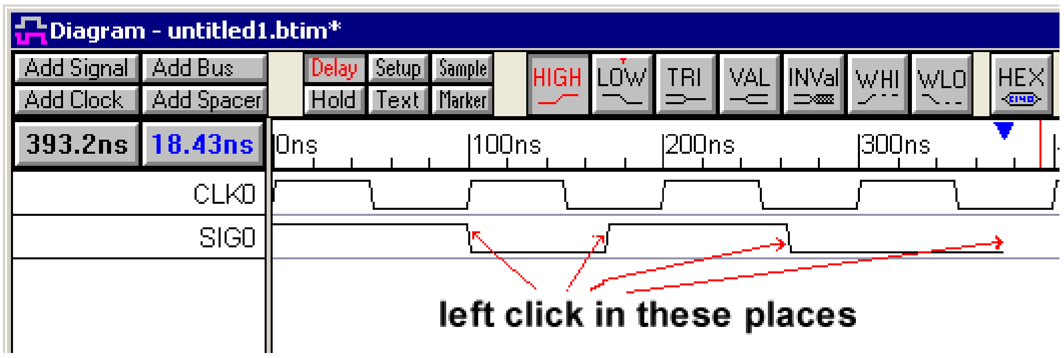
Duty Cycle %: 50 50

3) Add SIG0 and draw its waveform

- Click the **Add Signal** button to add a blank signal with the default name of SIG0.
- Look at the waveform buttons. The red one controls the type of waveform that is drawn next. After that the buttons will toggle to the state with the red T on top. Pressing the buttons will set a different draw and next state. This shows a toggling between high and low states.



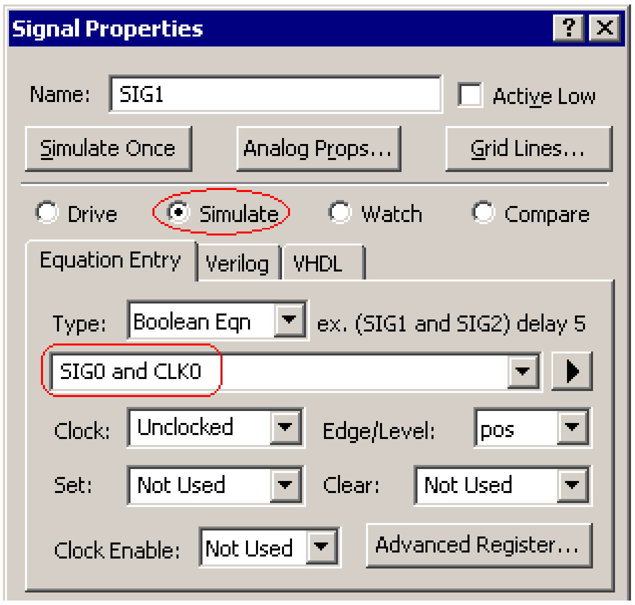
- Put the mouse cursor on the same row as the signal **SIG0** at about **100ns** and **left click** to draw a high waveform from 0ns to the mouse cursor position. Move the cursor to **150ns** and **click** the mouse button to draw a low segment from the end of the signal to the cursor point. Draw some more segments on SIG0 while watching the waveform buttons. Chapter 1: Signals and Waveforms 12 covers all of the signal drawing and display features.



4) Add signal SIG1 and simulate it as the a Boolean equation of SIG0 and CLK0

This step requires at least WaveFormer Pro. If you are using Timing Diagrammer skip to the next step.

- Click the **Add Signal** button to add a signal, then double click on the **SIG1** name to open the *Signal Properties* dialog. This dialog allows the name and other signal properties to be changed.
- Enter the equation **SIG0 and CLK0** into the Boolean Equation edit box. The signal names are case sensitive.
- Select the **Simulate** radio button. This setting automatically re-simulates SIG1 whenever an input waveform is changed.
- Click **OK** to close the dialog.


Signal Properties

Name: SIG1 Active Low

Simulate Once · Analog Props... · Grid Lines...

Drive [✓ Simulate] Watch Compare

Equation Entry | Verilog | VHDL

Type: Boolean Eqn ex. (SIG1 and SIG2) delay 5

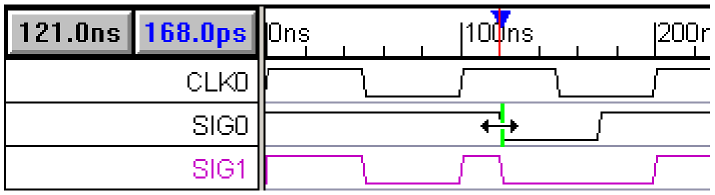
SIG0 and CLK0

Clock: Unclocked Edge/Level: pos

Set: Not Used Clear: Not Used

Clock Enable: Not Used Advanced Register...

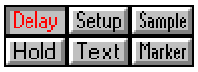
- Experiment with the simulated signal by dragging and dropping an edge on **SIG0**. Each time the edge is moved, a simulation is performed and **SIG1** redraws itself. (The clock edges cannot be moved with the mouse).



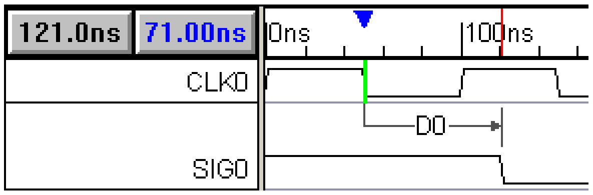
- Add a 10ns delay to the SIG1 equation by changing the equation to **(SIG0 and CLK0) delay 10**. Verify that SIG1 is correctly drawn. Chapter 4: Simulated Signals 59 covers the details on simulation.

5) Add a delay parameter named D0.

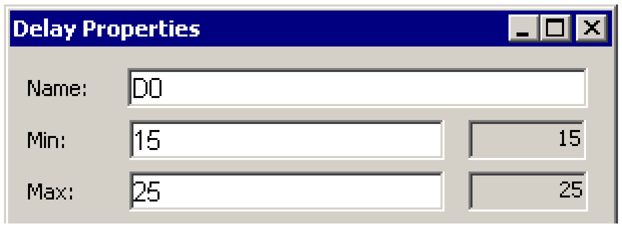
- Press the **Delay** button so that right clicks will add delays.



- Left click on the falling edge of **CLKO** at **50ns** to select the edge, then Right click on **SIG0's *first transition*** (at about 100ns). This draws a delay parameter from the selected clock edge to the SIG0 edge.

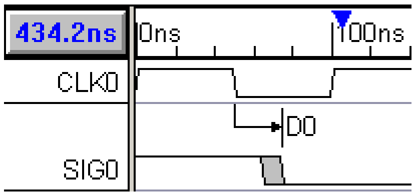


- Double-click on **D0** to open the *Delay Properties* dialog.
- Enter **15** into the **Min** box, and **25** into the **Max** box, then press **OK** to close the dialog.


Delay Properties

Name:	D0	
Min:	15	15
Max:	25	25

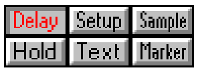
- Notice that the **SIG0** transition has moved so that it starts 15ns from the clock edge and the gray uncertainty region is 10ns wide (25-15=10). Chapter 5: Delay, Setup, & Hold Parameters 75 covers parameters in more detail.



Summary

Congratulations you have completed the Ultra-Quick Tutorial. The Tutorial covered starting an new timing diagram, generating a clock, drawing a signal, simulating a signal, and adding a delay parameter. In general, the following mouse functions are used in the program:

- **Left mouse** click to draw waveforms and move things around.
- **Right mouse** click to add the object highlighted in second group of buttons on button bar (like adding delays or text).



- **Double-click** on objects to edit them (like signal names, text objects, parameters, edges, clocks).

Please take the time to look through the manual to get an idea of the timing analysis and display features that are not covered in the Ultra-Quick Tutorial. Also the **Help > Tutorials** menu has much longer and in depth tutorials.

Chapter 1: Signals and Waveforms

This chapter covers how to add basic signals, draw the waveforms for the signals, and how to manage the display and navigate through large numbers of signals.

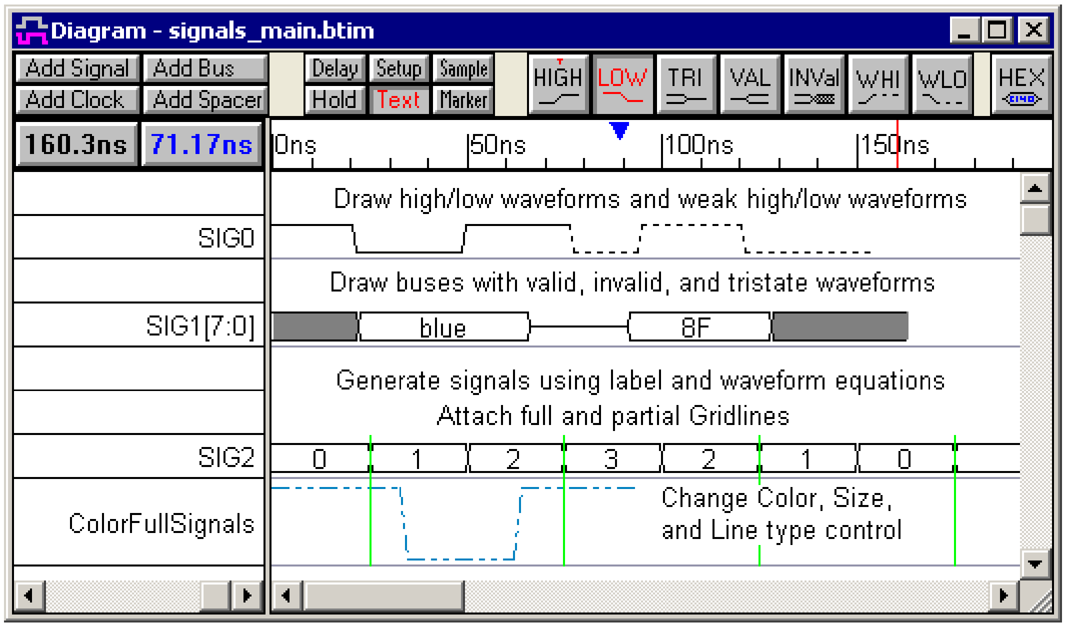


Diagram - signals_main.btim
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
HIGH
LOW
TRI
VAL
INVal
WHI
WLO
HEX
160.3ns
71.17ns
0ns
50ns
100ns
150ns
Draw high/low waveforms and weak high/low waveforms
SIG0
Draw buses with valid, invalid, and tristate waveforms
SIG1[7:0]
blue
8F
Generate signals using label and waveform equations
Attach full and partial Gridlines
SIG2
0
1
2
3
2
1
0
ColorFullSignals
Change Color, Size,
and Line type control

Later chapters cover more advanced signal types, which calculate their own waveforms or display specialized information:

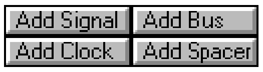
- **Clocks**, repetitive waveforms are drawn automatically, are covered in Chapter 2: Clocks 39.
- **Buses** and **Differential Signals**, multi-bit signals, are covered in Chapter 3: Buses 51.
- **Simulated Signals** which calculate their waveforms based on Boolean equations, registered logic, and behavioral HDL code are covered in Chapter 4: Simulated Signals 59.

1.1 Adding Signals and Signal Properties

Signals can quickly be added using the the appropriate button below. Double clicking on the name of a signal opens the *Signal Properties* dialog which sets the properties for a specific signal including it's name, export type, display features and whether or not the waveform is generated by a Boolean equation. This dialog can also edit several signals simultaneously.

To add a new signal:

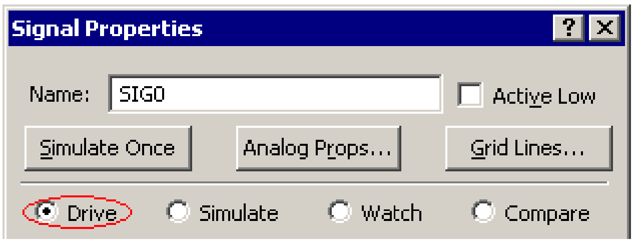
- Click the **Add Signal** button.



- Note: The above buttons are located in a timing diagram window. You can open a new timing diagram window using the **File > New Timing Diagram** menu.

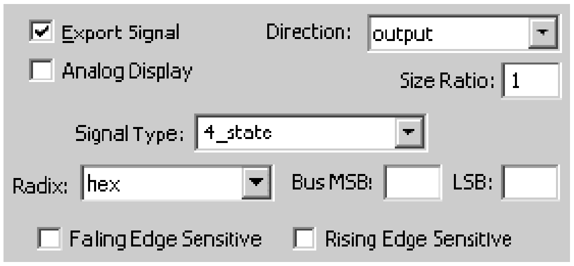
Edit a signal's name and other properties:

- Double-click on a ***signal name*** to open the *Signal Properties* dialog and type in a new name.
- **Active Low** signals are displayed with a bar (line) drawn over the top of the name. In equations, active low signals are referred to as ***name*$BAR.**

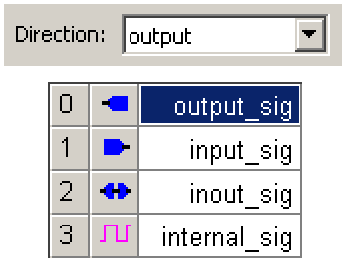


- **Drive** is the default state used for drawn signals and is the most common signal state for signals in a timing diagram. The other states are used by special features: **Simulate** is for creating continuously simulated signals (see Section 4.1: Boolean Equations with Delays 59), **Compare** is for waveform comparison signals (see Section 9.1: Performing a Signal Compare 130), and **Watch** is used by the BugHunter environment for watching signals in a simulation (see Section 2.2: Watching Signal and Component Waveforms in the BugHunter manual for more information).

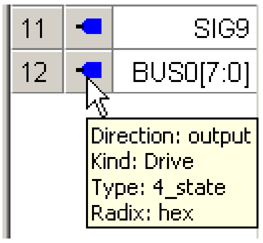
- **Radix** determines the base in which signal values are displayed.
- **Bus MSB** and **LSB** determines the bit size of the signal. A single bit signal is (0,0). Chapter 3 51 covers buses and differential signals.



- **Direction** sets the port direction of the signal relative to the testbench. Output signals drive the model under test. Input signals represent signals with data coming back from the model under test. Internal signals are signals that are not connected to the ports of the model under test. The direction is also shown as an icon to the left of the signal name in the label window. To turn off this column choose the **Options > Draw Preferences** menu and uncheck the **Show Direction Icons** box.



- A quick way to view a signal's properties is to put the mouse over the direction column and a tooltip will appear. Placing the mouse over the signal's name shows a tooltip of the full signal name, which is handy for signals with long names.



Signal Name Rules: Normally spaces are not allowed in signal names. For documentation purposes, you can get spaces into a signal name by using the **Edit > Search and Rename** menu. However, adding spaces to signal names may cause problems when exporting to different formats or simulating Boolean Equations. Signal names beginning with "**$$**" are reserved for internal use. Signal names should not be the same as common Boolean operators (examples: AND, OR), if you plan to export to VHDL.

To move or sort signals:

- Section 1.11: Selecting, Moving, and Sorting Signals 36 describes several techniques for moving signals with the mouse or sorting them into a particular order. The easiest way to move a signal is to select the signal label and drag it using the mouse. Also, the **Edit > Sort Signals By Name** menu will rearrange signals in alphabetical order.

To delete a signal or multiple signals:

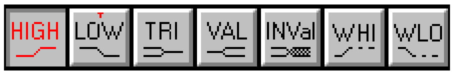
- Click on one or more ***signal names*** to highlight the names and then press the **Delete** key. When a signal is deleted, any parameters or text connected to it are also deleted. The **Edit > Undo** or **Edit > Undo Delete** menu item will undo a mistaken delete.

Change the Auto Name generation:

- By default, signals are named SIG# but the "SIG" prefix can be changed by right clicking on the **Add Signal** button to open the *Modify Auto Signal Name Prefix* dialog and entering a new prefix. Bus and Clock prefix names work the same way.

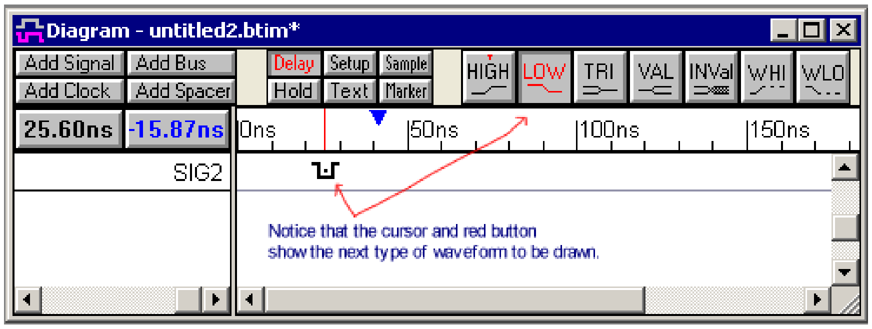
1.2 Drawing Waveforms

The timing diagram editor is always in drawing mode so left clicking on a signal will draw a waveform. The red state button controls the type of waveform that is drawn (high, low, tri-state, valid, invalid, weak high, and weak low). The buttons toggle back an forth between two states, and the next state is indicated by the little red T on top. Click on the state buttons to set the toggle and next state.



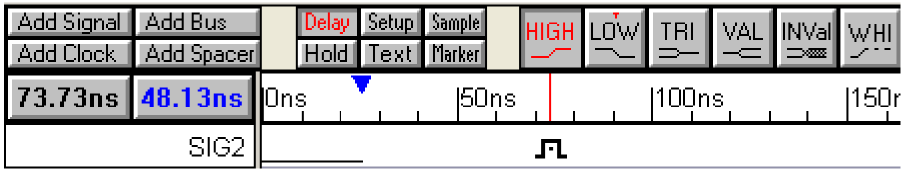
To draw the waveform of a signal:

- Place the mouse cursor inside the *Diagram* window at the same vertical row as the signal name.The red state button on the button bar determines the type of waveform drawn. The cursor shape also mirrors the red state button.

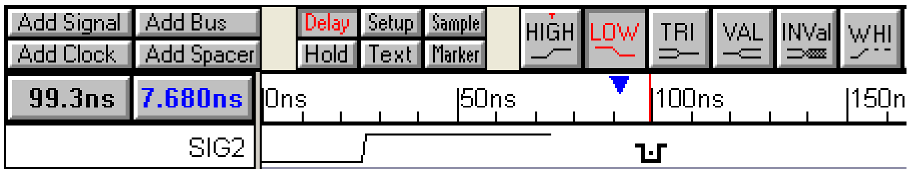


Notice that the cursor and red button show the next type of waveform to be drawn.

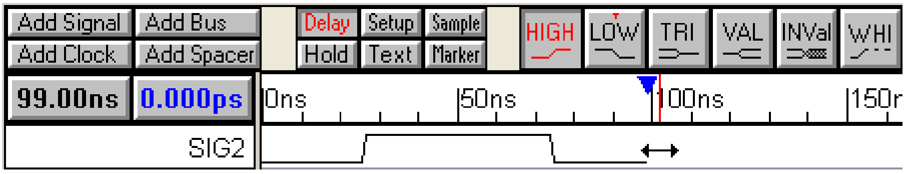
- Click the left mouse button. This draws a waveform from the end of the signal to the mouse cursor.



- Move the mouse to the right and click again to draw another segment.



- Keep drawing from left to right across the diagram.

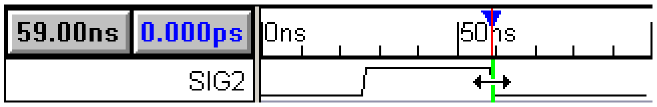


- Pressing the **middle mouse button** either toggles the state buttons or cycles through them depending on the setting in the *General Preferences* dialog. Choose **Options > General Preferences** menu to open the dialog.

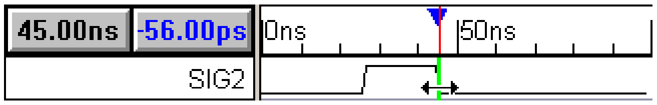
There are several mouse-based editing techniques used to modify existing waveforms. These techniques will only work on signals that are drawn. They will not work on generated signals like clocks and simulated (blue) signals that are covered in later chapters.

1) Drag-and-Drop a Signal Transitions:

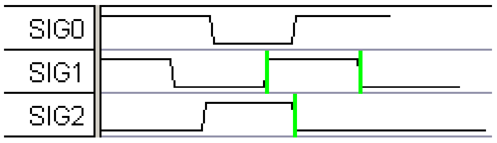
- To move one transition, click on the transition and drag it to the desired location.



- To move all of the transitions on one signal, hold down the **<1>** and **<2>** number keys while dragging. Holding down just the <1> key moves all the edges to the left, and the <2> key moves all the right.

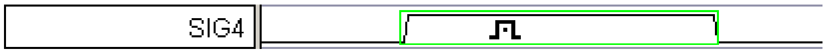


- To move transitions on different signals, first select the transitions by holding the **<CTRL>** while clicking on them. Then select and drag on the final selected edge to move all the selected transitions an equal distance in time.

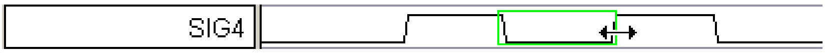


2) Click-and-Drag to insert a segment into a waveform:

- Inside of a segment, click and drag the cursor to insert a segment

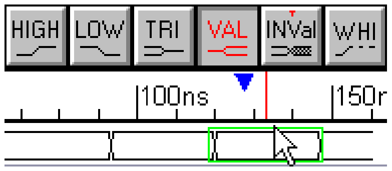


- The inserted state is determined by the red state button

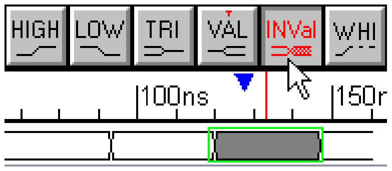


3) Change a segment's graphical state by selecting it and then pressing a state button:

- Click in the middle of the segment to select it (so that it has a green box around it).



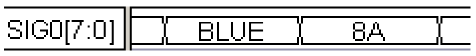
- Click on a state button to apply that graphical state to the segment. If you change a segment to same level as an adjacent section, the transition will turn red to preserve the edge data. This transition can be deleted if necessary.



4) Adding virtual state Information to a segment

Chapter 3.1: Virtual Buses 51 covers all of details of Virtual states on buses.

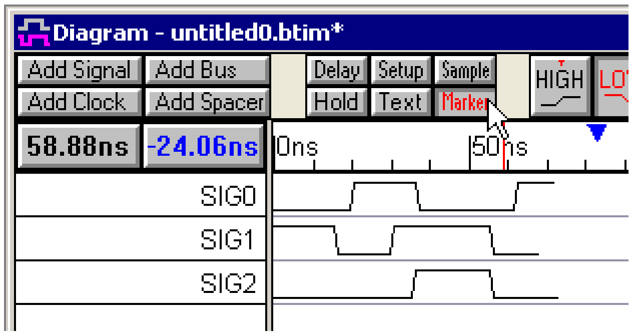
- For Signals, double-click on the middle of a segment to open the *Edit Bus State* dialog, and then type in a new value into the **Virtual** edit box.



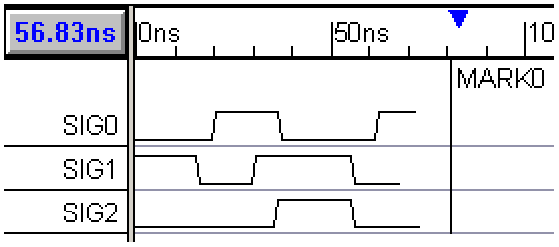
- For Clocks, press the **Hex** button and then double-click on the middle of the segment to open the *Edit Bust State* dialog. If the Hex button is not pressed, the double-click will open a different dialog to allow editing of the clock.

5) Making Waveforms end at a common time

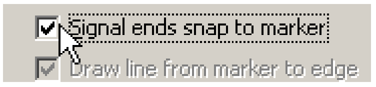
- Sketch waveforms, then press the **Marker** button down.



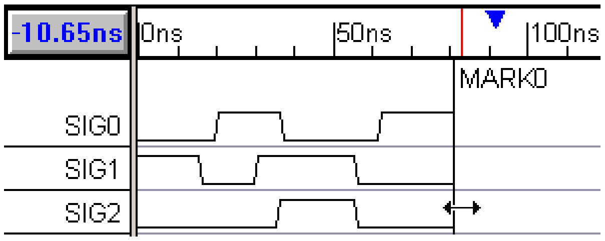
- Right click in the diagram window past the ends of the waveforms to add a time marker line.



- Double-click on the time marker line to open the *Edit Time Marker* dialog, and check the **Signal ends snap to marker** control.



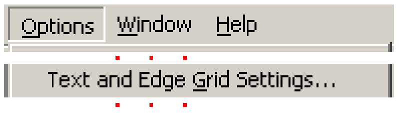
- Note that the waveforms snap to the marker, but are not truncated. All signal transitions remain intact, even if the marker is moved. Only the last segment of the signal ends at the marker



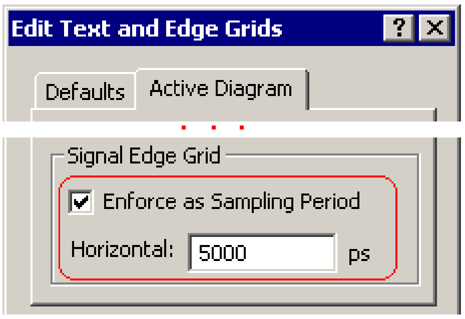
5) Adjusting the drawing Grid

Drawn signals transitions are automatically aligned to the closest grid time. The grid does not affect the placement of edges that are moved by delays or formulas. By default the grid is set to the base time unit, because this generates nice VHDL and Verilog stimulus generation files with whole number times (like 2ns instead of 2.465ns). 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.

- Choose the **Options > Text and Edge Grid Settings** menu item to open the *Edit Text and Edge Grids* dialog.



- The **Signal Edge Grid** section controls where the waveform edges will be placed on the **Horizontial** grid.
- The **Enforce as Sampling Period** control is used by the Analog State Label equations and does not effect the drawing of signals (see Section 8.3: Sine, Capacitor, Ramp & Exponential waveforms 120). A newer method of generating analog waveforms that are editable as equations are described in Section 8.2: Waveform Equation Blocks for editable Analog waveforms 116.



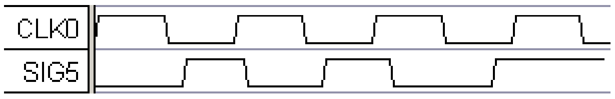
Enforce as Sampling Period: ✓

Horizontal: 5000 ps

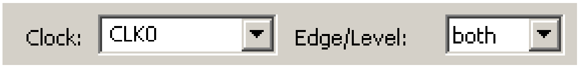
7) Snap signal transitions to the closest clock edge

A quick way to line up the edges of signal is to snap them to the previous clock edge using the **Cyclize Selected Signal(s)** context menu.

- Sketch a waveform that is close to lining up with a clock.

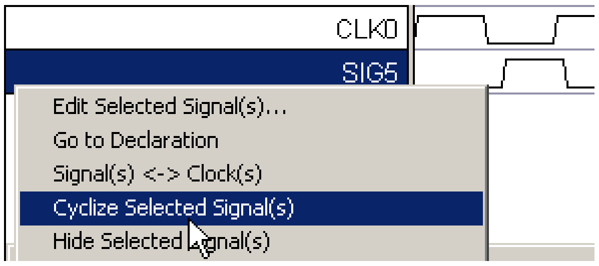


- Setup the signal's clocking signal, by double clicking on the signal name to open the *Signal Properties* dialog, and set he clocking signal using the **Clock** box and set the **Edge/Level** to **both** (or just neg or pos if you only want to snap to those edges).

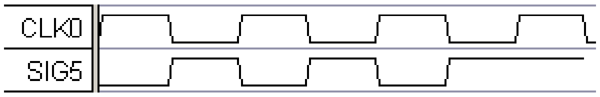


Clock: CLK0 Edge/Level: both

- Right-click on the signal name and choose **Cyclize Selected Signal(s)** from the context menu.

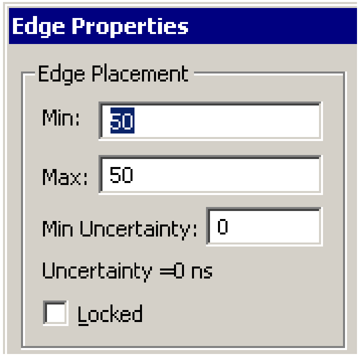


- Notice that the edges are lined up with the clock edges



8) Finding the exact edge time and locking the edge

- Double-click on an edge of the signal transition to open the *Edge Properties* dialog. Section 1.5[26] discusses timing analysis features of this dialog.
- To move an edge, enter a new **min** or **max** time.
- To lock an edge so that it cannot be moved, check the **Locked** checkbox.



Edge Properties

Edge Placement

Min: 50

Max: 50

Min Uncertainty: 0

Uncertainty =0 ns

☐ Locked

- Note: All edges on a signal can be locked by selecting the signal name, and then choosing the **Edit > (Un)Lock Edges of Selected Signals** from the main menu.

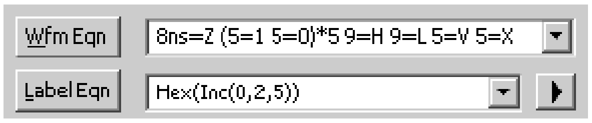
1.3 Generating Waveforms and States with Equations

Waveforms can be generated from Waveform Equations and automatically labeled using State Label equations. These features provide a quick way to generate signals that have a known pattern that is more complicated than a periodic clock cycle. The label equations also provide a quick way to define analog data for a waveform.

This is an older method of generating waveform data and these equations generate "one-time" results (the generated events react thereafter the same as if you had manually drawn these events). A newer method called Waveform Equation Blocks preserves the original equation for subsequent editing. These are described in Section 8.2: Waveform Equation Blocks for editable Analog waveforms[116].

To apply a Waveform or Label Equation to a signal:

- Double-click on a signal's name to open the *Signal Properties* dialog.
- Enter an equation and press the **Wfm Eqn** or **Label Eqn** button to apply it.



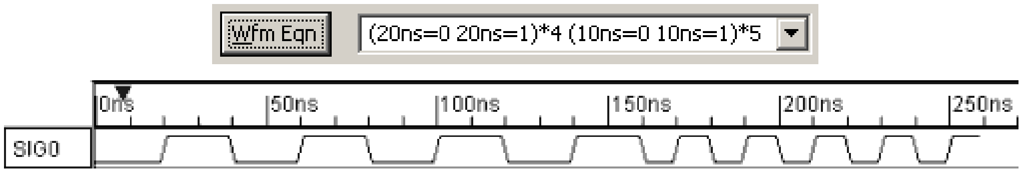
Wfm Eqn: 8ns=Z (5=1 5=0)*5 9=H 9=L 5=V 5=X

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

- The drop-down boxes have example equations and a history of past equations that you can modify so that you do not have to memorize the syntax. The black triangle is a quick add box that will enter the function values into the box.

Rules for Waveform Equations:

Each time you press **Wfm Eqn** button the waveform equation will be appended to the end of the signal, so you can build up a waveform using several smaller equations. The following example, uses a waveform equation to generate a variable frequency clock that switches from 25Mhz to 50Mhz. The 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 waveform equation. Pressing the button again would apply another set of waveforms to the end of the signal.

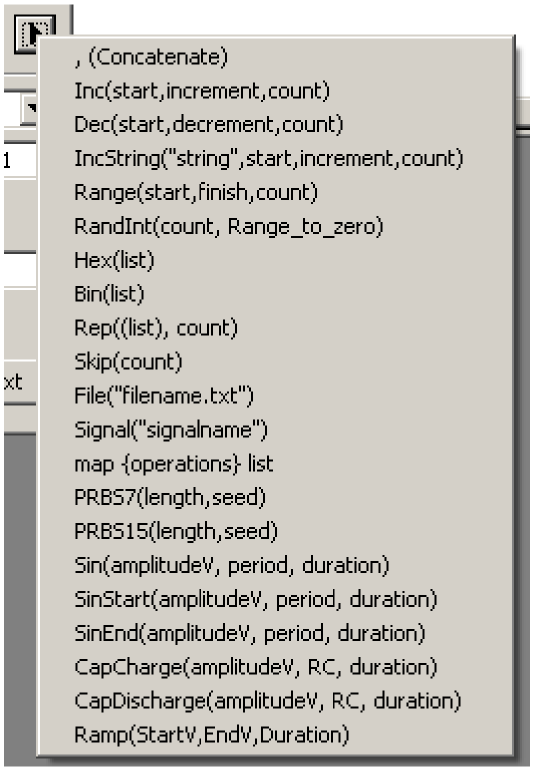


- A Waveform equation is a list of ***space-separated***, time-value pairs in the form **TimeValue [units]=StateValue** or **TimeValue=StateValue**.
- If **[units]** are not specified, then the "display time unit" of the current project will be used.
- Legal graphical **StateValues** are: 0 = strong low, 1= strong high, Z = tristate, V = valid, X = invalid, L = weak low, and H = weak high.
- If the **StateValue** is not a graphical state, then a valid segment will be created and labeled with the state value. This allows you to label a signal with bus values and symbolic names while using the Waveform Equations. For example, **10ns=hello** will draw a 10ns long valid segment with a state value of *hello*.
- For loops, enclose a list of time-value pairs in ***parenthesis*** followed by a ***multiply symbol*** and the number of times the list is to be repeated.
- The default waveform equation contains all of the available states and syntax, so If you start by editing the default equation then you do not have to memorize the waveform equation rules.

Rules for Label Equations:

Each time you press the **Label Eqn** button the old state values are discarded and the new ones are written in (unlike the waveform equations which append to the end). Label equations provide a quick way to enter state values into digital counter signals and analog sine and capacitor discharge signals. For example a simple counter that starts at one and counts to ten could be written as: `Skip(3),Inc(1,1,10)`. This tells the program to skip three segments, then to start labeling at one and place ten labels, incrementing by one each time. Functions can be combined by using a **comma** between the commands.

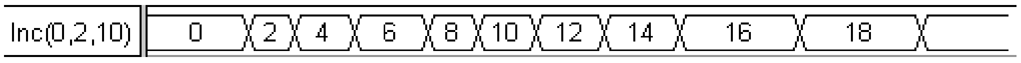
- The black triangle to the right of the box is a quick fill box that will enter an equation into the state label box. Then all you have to do is replace the variables with your values.
- Tip: First use a Waveform Equation to create an unlabeled signal with the correct number of segments, and then label it with a State Label Equation. For example (20=V)*100 creates a waveform with 100 valid segments that are 20ns wide.
- A Label equation is a ***comma separated*** list of label functions.
- The label functions are applied in the order listed.
- Each label function returns a value of **list** so whenever you see **list** you can use a function, a single value like 5 or blue, or a list in parentheses (2,3,4).


, (Concatenate)
Inc(start,increment,count)
Dec(start,decrement,count)
IncString("string",start,increment,count)
Range(start,finish,count)
RandInt(count, Range_to_zero)
Hex(list)
Bin(list)
Rep((list), count)
Skip(count)
File("filename.txt")
Signal("signalname")
map {operations} list
PRBS7(length,seed)
PRBS15(length,seed)
Sin(amplitudeV, period, duration)
SinStart(amplitudeV, period, duration)
SinEnd(amplitudeV, period, duration)
CapCharge(amplitudeV, RC, duration)
CapDischarge(amplitudeV, RC, duration)
Ramp(StartV,EndV,Duration)

Digital state label functions

Inc(start, increment, count)

This function generates labels that begin at **start** and increase by **increment** each time. A total of **count** labels will be generated. The actual labels generated will be *start, start + increment, start + (increment * 2), …, start + increment * (count - 1)*.

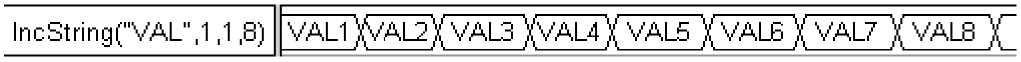


Dec(start, increment, count)

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

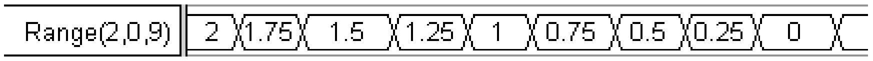
IncString("string", start, increment, count)

This function behaves identically to the **Inc** function, except that the incremented label values are appended to the **string**.



Range(start, finish, count)

This function generates **count** state labels, evenly distributed over the range of values from **start** to **finish** inclusive.



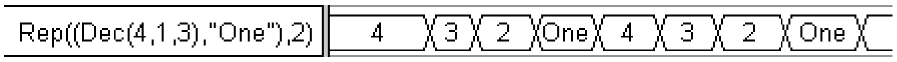
RandomIntegerArray(count, Range_to_zero) or RandInt(count, Range_to_zero)

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

Range_to_zero.

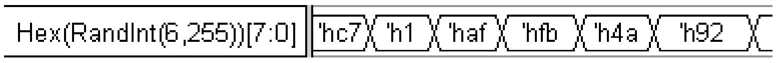
Rep((list), count)

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**). If the list contains words, place them in quotes. Note that the list must be placed in its own pair of parentheses.



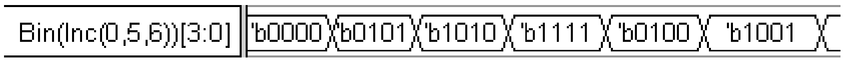
Hex(list)

This function formats the items in **list** as hexadecimal. 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.



Bin(list)

This function formats the items in **list** as binary. All numbers in the list are converted to a base-2 representation. The numbers are prefixed by **'b** to indicate that they are binary. 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.



Skip(count)

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

File("*filename.txt*")

This function imports a set of state values from the specified text 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.

Signal("*signalname*")

This function copies the states from the signal indicated by *signalname* and applies them to the current signal. 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).

map{operations} list

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 curly brackets **{ }**. The second argument is the list whose elements are to be operated on. The special variable **$_** is used to represent the state value to be operated on (it takes on the value of each state in the list). Once you become familiar with it, the **map** function in combination with standard Perl language operations gives you extraordinary expressive power. See Chapter 15 [191] for more information on Perl. For example the **map{$_ + 5} Signal()**. The **Signal** function returns a list of the states on the current signal, and the **map** function increments each state in the array by 5.

Analog state label functions

Analog state label equations are covered in Section 8.3 Sine, Capacitor, Ramp, and Exponential waveforms 120 and covers **Sin**, **SinStart**, **SinEnd**, **CapCharge**, **CapDischarge**, **Ramp**, and **Exponential**. This is an older method of generating analog waveform data and these equations generate "one-time" results (the generated events react thereafter the same as if you had manually drawn these events). Unlike the Waveform Equation Blocks described in Section 8.2: Waveform Equation Blocks for editable Analog waveforms 116, the original equation cannot be changed to change the waveform, only the individual events can be edited.

User-defined state label functions

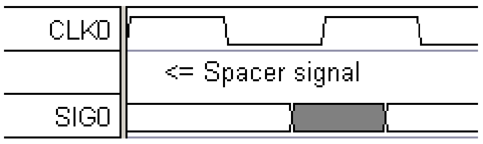
Users can also add new functions by placing them into the perl script **labelequation.pm** located in the **SynaptiCAD > Perl** directory. Normally, state label equations are entered using the *Signals Properties* dialog, but they can also be placed in a perl script. Perl scripts are useful for creating especially complex waveforms. The file **wfm_analog_example.pl** in the perl directory has an example of using a label equation inside of a Perl script.

1.4 Colors, Line Type, Size, and More Visual Controls

There are many ways to control how signals and waveforms are displayed including color, font, line thickness, edge slant, and signal spacing.

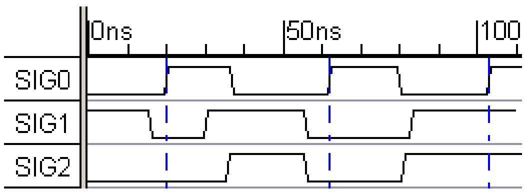
Insert Spacer Signals:

- Press the **Add Spacer** button to insert a horizontal space into the diagram below the selected signal.



Grid Lines on Signals:

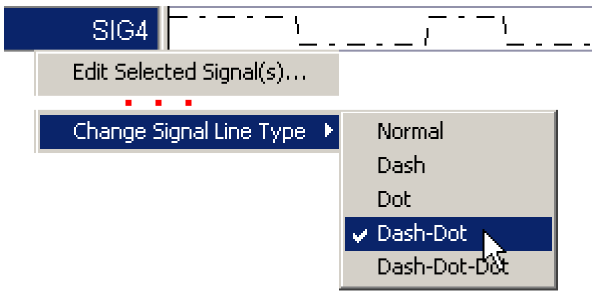
- Double click on a signal name to open the *Signal Properties* dialog, then press the **Grid Lines** button to open the *Grid Options* dialog (see Section 2.3 Grid Lines on Clocks and Signals 44).



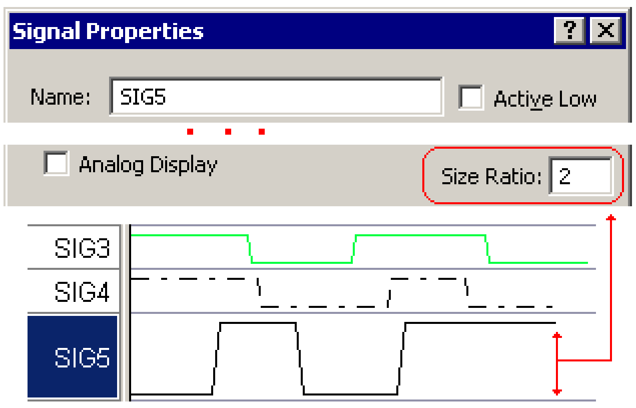
- The grid lines will still be visible if the controlling signal is hidden and the **View > Show Hidden Text** menu is checked. This lets you create patterns of grid lines without having the attached signal disturbing the circuit information.

Individual Waveform Settings with right click menus and Signal Properties dialog:

- **Line type:** right click on the signal name and choose **Change Signal Line type** to pick a different line type like Dashes or Dots. (Line thickness is a global controls shown below)

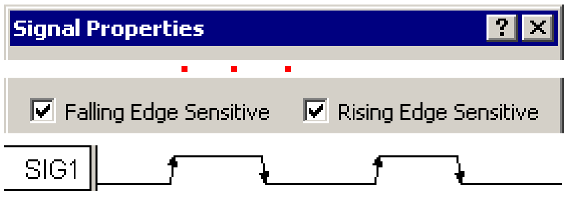


- **Height of one Signal:** double click on the signal name to open the *Signal Properties* dialog and set the **Size Ratio** to increase the height of the signal.
- **Height of all Signals:** Select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog. The **Waveform Height** sets the height of all of the waveforms.



Name: SIG5
☐ Active Low
☐ Analog Display
Size Ratio: 2

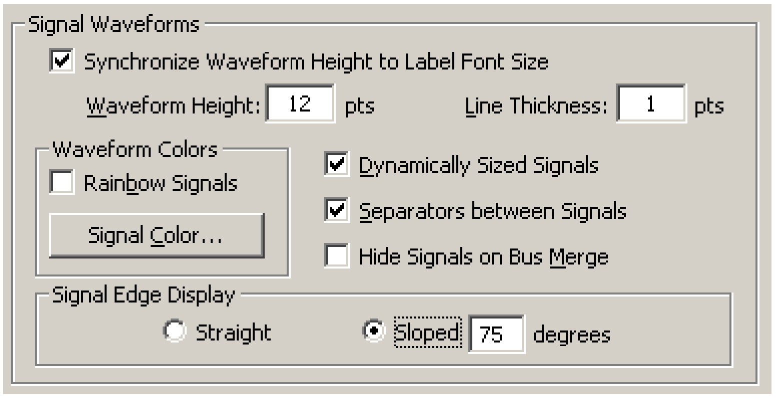
- **Arrows on waveform edges:** double click on the signal name to open the *Signal Properties* dialog and check the **Falling Edge Sensitivity** or **Rising Edge Sensitivity** boxes. These are also used by TestBencher Pro for sequence recognition.



☑ Falling Edge Sensitive ☑ Rising Edge Sensitive

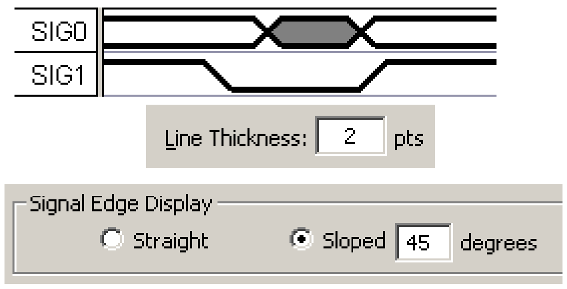
Global Signal Waveform Controls using the Drawing Preferences Dialog:

- Select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog and look at the **Signal Waveforms** section.
- **Synchronize Waveform Height to Label Font Size** keeps the **waveform height** relative to the height of signal label font.



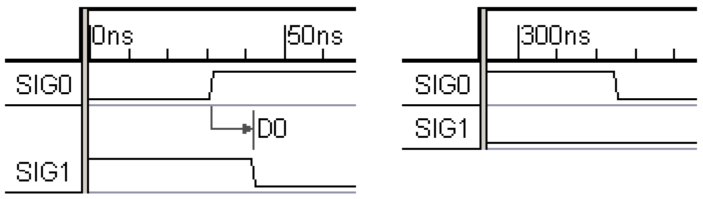
☑ Synchronize Waveform Height to Label Font Size
Waveform Height: 12 pts
Line Thickness: 1 pts
☐ Rainbow Signals
☑ Dynamically Sized Signals
☑ Separators between Signals
☐ Hide Signals on Bus Merge
○ Straight ● Sloped 75 degrees

- **Line Thickness** can be increased and the edges can be made **Straight** or **Slanted**. In the picture, the signals have a thickness of 2 and a slant of 45% (default is 75%).
- To make straight edges for just one signal, clock, or an Analog Signal, check the **use straight edges** in the *Analog Properties* dialog, see [Section 8.1 Analog Waveform Display](115).

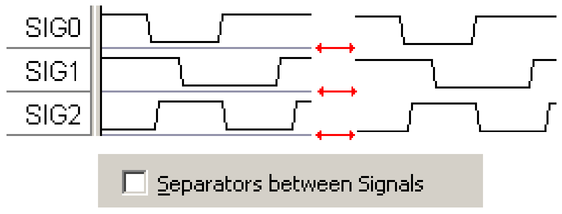


Line Thickness: 2 pts
○ Straight ● Sloped 45 degrees

- **Dynamically Sized Signals** , if checked, will allow the space between signals to automatically change height as parameters are scrolled off the screen.



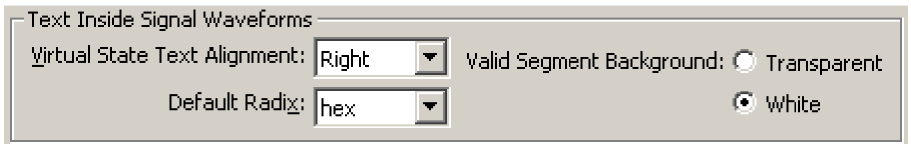
- Uncheck **Separators Between Signals** to remove the grey lines between signals.



- **Hide Signals on Bus Merge** hides the member signals when a group bus is created, see Section 3.2: Group Bus 53.

Global Controls that effect text inside signal waveforms using the Drawing Preferences Dialog:

- Select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog and look at the **Text Inside Signal Waveforms** section.



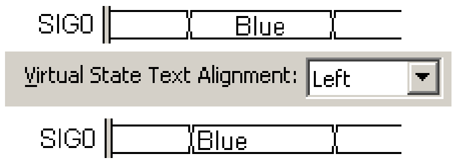
Text Inside Signal Waveforms

Virtual State Text Alignment: Right

Default Radix: hex

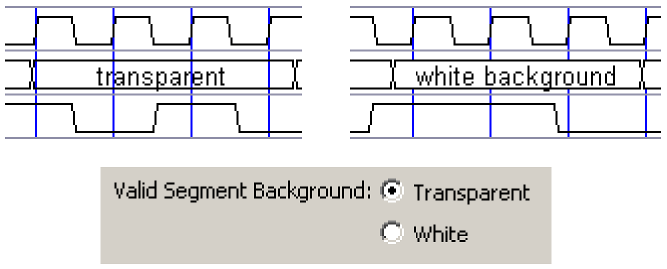
Valid Segment Background: ( ) Transparent [✓ White]

- **Virtual State Text Alignment** controls whether the virtual state text in a segment will appear centered, left-justified, or right-justified.



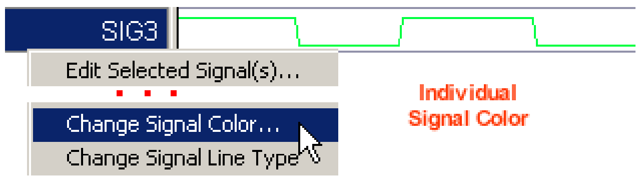
- **Default Radix** specifies the base in which signal values are displayed for all new signals. Individual control is in the *Signal Properties* dialog.

- In DataSheet Pro, the **Valid Segment Background** setting specifies whether the background color of a valid segment will over write other objects like markers and text objects.

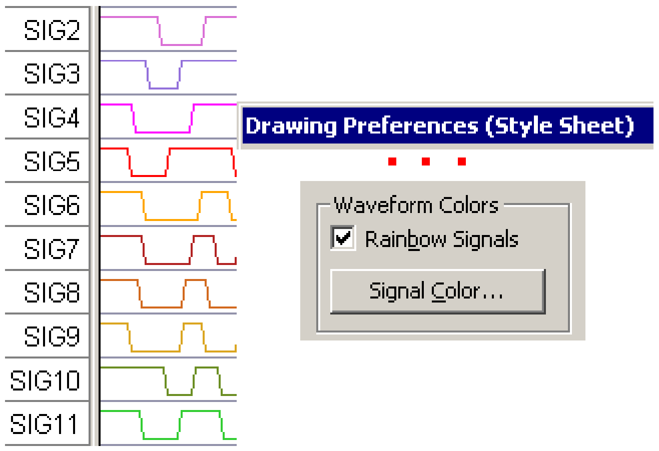


Color Controls for Waveforms and Signal Names

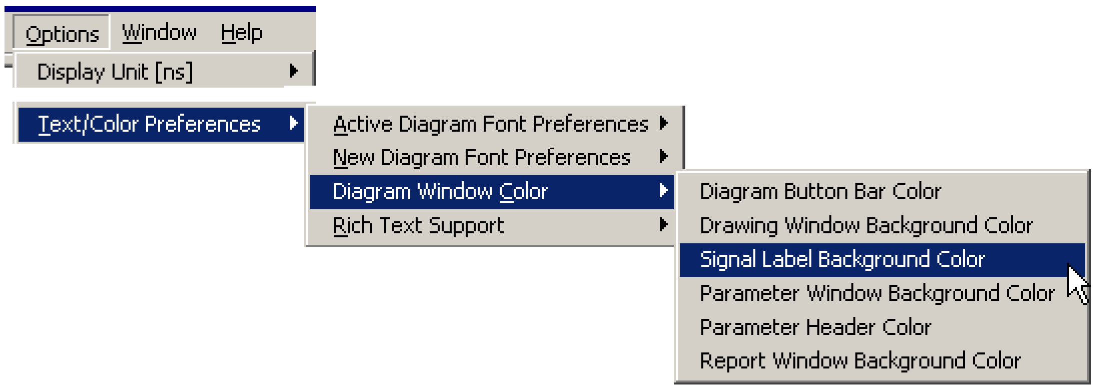
- **Color One Waveform:** right click on signal name and choose **Change Signal Color** . A global setting for all signals is shown below



- **Color All Waveforms:** Choose **Options > Drawing Preferences** to open the dialog.
    - **Rainbow Signals** changes the signals to multi-colored waveforms.
    - **Signal Color** changes the color of all of the signals in the diagram (this only works if rainbow signals is unchecked).



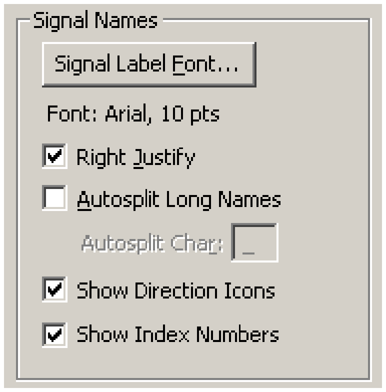
- **Color of Label Window and Drawing Window:** Choose the **Options > Text/Color Preferences** menu. There are several sub-menus that are used to set the different font and color properties. The **Active** menus change the current diagram. The **New** menus change the default settings for new diagrams.


Options
Window
Help
Display Unit [ns]
Text/Color Preferences
Active Diagram Font Preferences
New Diagram Font Preferences
Diagram Window Color
Rich Text Support
Diagram Button Bar Color
Drawing Window Background Color
Signal Label Background Color
Parameter Window Background Color
Parameter Header Color
Report Window Background Color

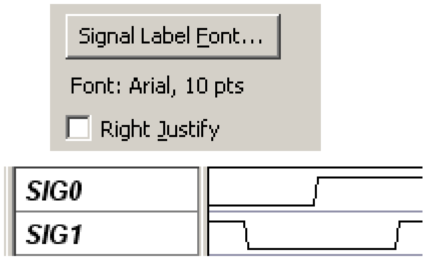
- Simulated signals by default are displayed using purple signal color, but they can be made to display using the normal signal color by unchecking the **View > Show Default Simulated Signals color** menu.

Global Signal Name Controls using the Drawing Preferences Dialog:

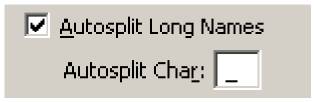
- Select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog and look at the **Signal Names** section.


Signal Names
Signal Label Font...
Font: Arial, 10 pts
[✓] Right Justify
[ ] Autosplit Long Names
Autosplit Char: _
[✓] Show Direction Icons
[✓] Show Index Numbers

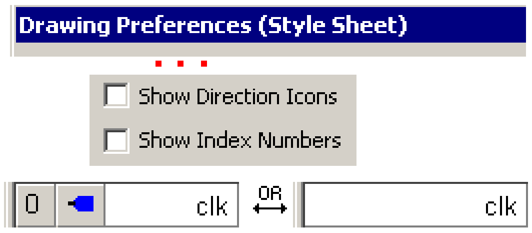
- **Signal Label Font** changes the font used for signal names.
- **Right Justify** causes the signals names to be lined up on the right side of the label window. Un-checking it causes it to be left justified.


Signal Label Font...
Font: Arial, 10 pts
Right Justify

- **Autosplit Long Names:** allows the program to split signal names that are too long for the display. They will be split at the **Autosplit Char**.

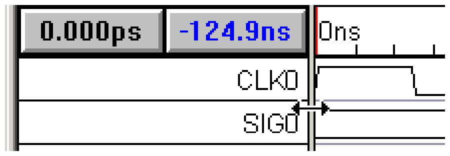

☑ Autosplit Long Names
Autosplit Char: _

- Hide the index and direction columns in the signal label window by unchecking the **Show Direction Icons** and **Show Index** boxes.


Drawing Preferences (Style Sheet)
Show Direction Icons
Show Index Numbers

Width of the Label window:

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



Change the default naming conventions of the Label window:

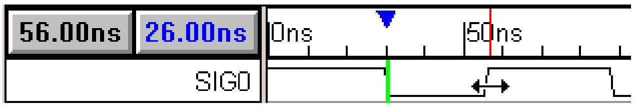
- Right-click on the **Add Signal** button to open the *Modify Auto Name Prefix* dialog.
- Enter the new prefix so that each new signal will be named ***Prefix#***, instead of the default ***Sig#*** .

1.5 Measuring Time and State values

There are several different ways to measure and display times and state values.

Time and Delta Buttons

- The **Time Button**, with the black writing, displays the current position of the mouse cursor in the *Diagram* window. The **Delta Button**, 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.
- To measure, left click on an edge (to select it and also move the delta mark), then move the mouse over another edge. The Delta button will

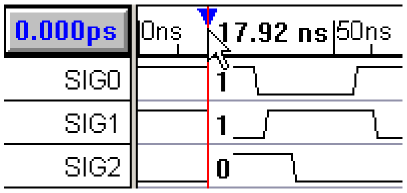


display the difference between the mouse position and the blue mark.

- To turn off the continuous measurement in the buttons, choose the **Options > General Preferences** menu to open a dialog and then uncheck the **Continuous Measurement** box.
- **Continuous Measurement:** 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.

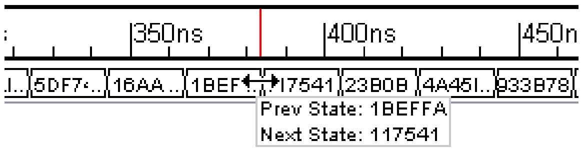
Display Signal States by clicking in the time line

- Click in the timeline in the *Diagram* window to drop a temporary marker line that displays the numerical state of each signal.



Display Signal States by hovering over a state transition or bus segment

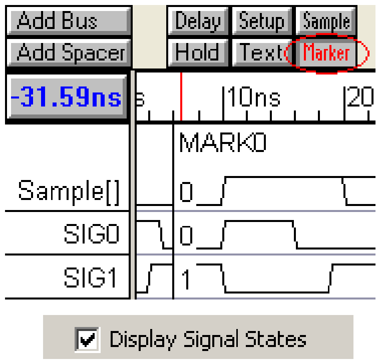
- The mouse tool-tip displays the next and previous states of bus segments when the bus hovers over a state transition. This feature can be disabled by unchecking the **View > Show State at Cursor** menu.



- Bus states automatically show partial extended state data when the segment is too small to show entire state in diagram window.

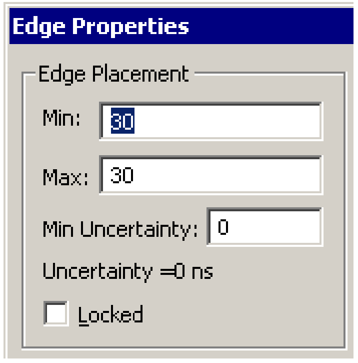
Display Signal States using Markers

- Select the **Marker** button and right click in the timing diagram to add a Marker. Hovering over a marker will show the state values of the signals at that time. To disable this feature, uncheck the **Popup Display Signal States for Markers** checkbox option in the *Drawing Preferences* dialog.
- To permanently display the state values beside a marker, double click on the marker and check the **Display Signal States** checkbox in the *Marker Properties* dialog.



Using the Edge Properties Dialog to Measure Times and States

- Double click on the edge to open the *Edge Properties* dialog which displays the min and max times and uncertainty. Entering a new time moves the edge.
- The **min uncertainty** edit box lets you add a minimum uncertainty region to an edge. 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.


Edge Properties

Edge Placement

Min: 30

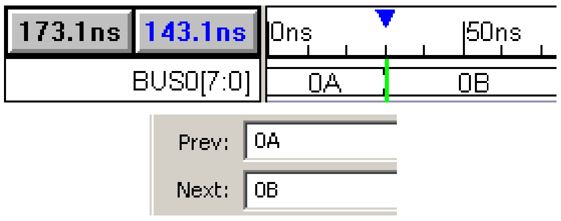
Max: 30

Min Uncertainty: 0

Uncertainty =0 ns

☐ Locked

- 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.
- The **Prev** and **Next** boxes display the states on the buses. This is particularly useful when the states are very close together.



Prev: 0A

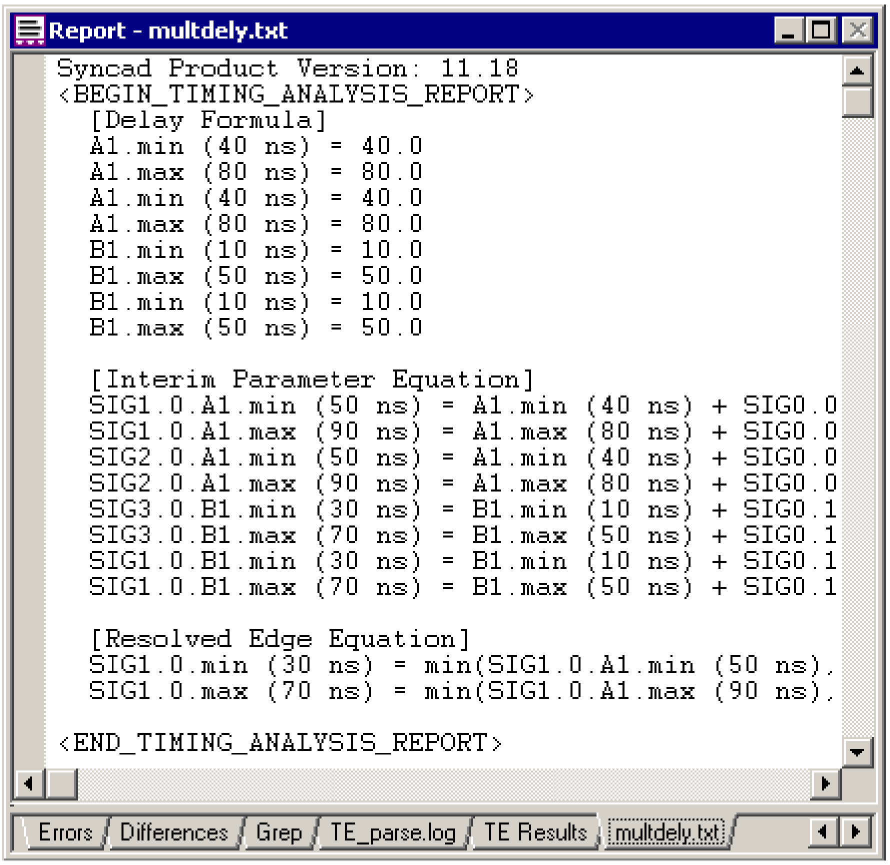
Next: 0B

- The **Prev** and **Next buttons** moves the dialog to the previous or next edge on the signal.

Generate a Timing Analysis Report

To figure out how the time for a particular edge is calculated, you can generate a Timing Analysis Report.

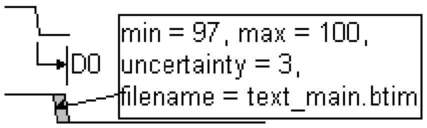
- Choose **File > Save Timing Diagram As** and select **Timing Analysis Report** from the *Save As Type* box.
- When the file is saved, it will also be opened in a tab in the Report window. If you cannot see the Report window, choose the **Window > Report**.

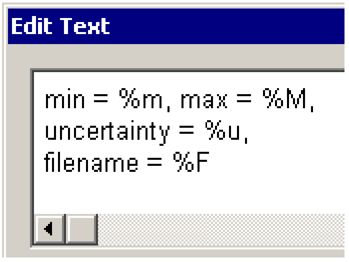


```
Syncad Product Version: 11.18
<BEGIN_TIMING_ANALYSIS_REPORT>
  [Delay Formula]
  A1.min (40 ns) = 40.0
  A1.max (80 ns) = 80.0
  A1.min (40 ns) = 40.0
  A1.max (80 ns) = 80.0
  B1.min (10 ns) = 10.0
  B1.max (50 ns) = 50.0
  B1.min (10 ns) = 10.0
  B1.max (50 ns) = 50.0

  [Interim Parameter Equation]
  SIG1.0.A1.min (50 ns) = A1.min (40 ns) + SIG0.0
  SIG1.0.A1.max (90 ns) = A1.max (80 ns) + SIG0.0
  SIG2.0.A1.min (50 ns) = A1.min (40 ns) + SIG0.0
  SIG2.0.A1.max (90 ns) = A1.max (80 ns) + SIG0.0
  SIG3.0.B1.min (30 ns) = B1.min (10 ns) + SIG0.1
  SIG3.0.B1.max (70 ns) = B1.max (50 ns) + SIG0.1
  SIG1.0.B1.min (30 ns) = B1.min (10 ns) + SIG0.1
  SIG1.0.B1.max (70 ns) = B1.max (50 ns) + SIG0.1

  [Resolved Edge Equation]
  SIG1.0.min (30 ns) = min(SIG1.0.A1.min (50 ns),
  SIG1.0.max (70 ns) = min(SIG1.0.A1.max (90 ns),

<END_TIMING_ANALYSIS_REPORT>
```

Display time using Text objects, Parameters, or Markers

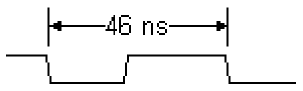
- **Use a Text Object with control codes to display edge times:** press the **Text** button, select the edge, then right-click to open an edit box. Enter **%m**, **%M** , or **%u** control codes to display min transition, max transition, or uncertainty region (Section 6.1: Adding Text 91).





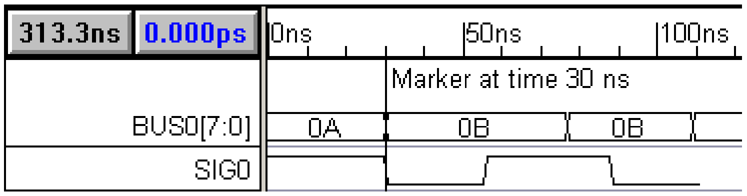
```
min = %m, max = %M,
uncertainty = %u,
filename = %F
```

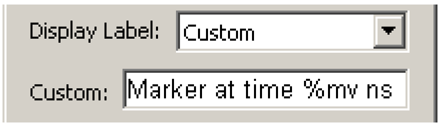
- **Display the distance between two transitions:** Add a setup or hold parameter between the two transitions. Double-click on the parameter to open the *Parameter Properties* dialog, and choose the **Distance** from the Display label drop-down, and optionally check the **Outward arrows** box (Section 5.3 Display Settings for Parameters 80).



- **Display the exact placement of a time marker**: Press the **Marker** button the right-click in the diagram to add a time marker line. Double-click on the marker line to open the *Edit Time*

Marker dialog and change the Display label to **min value** or **custom** using the **%mv** control code (Section 6.4 Markers Lines 96).





Display Label: Custom

Custom: Marker at time %mv ns

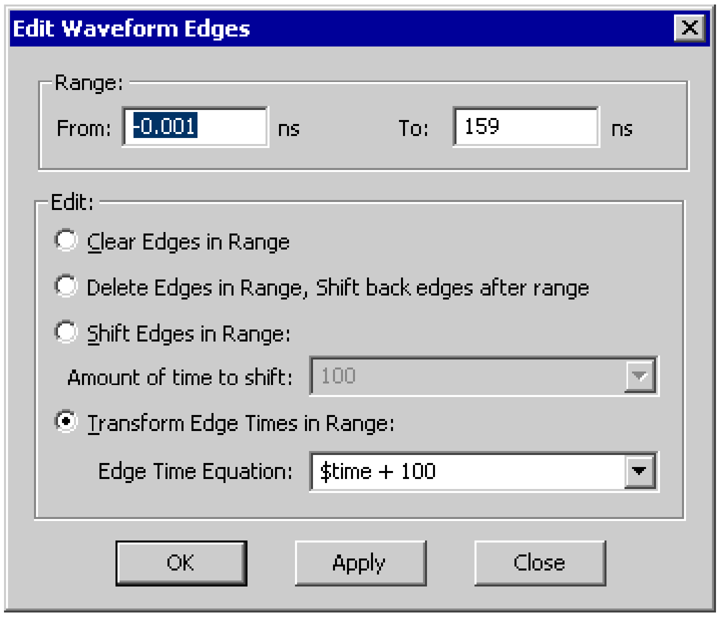
1.6 Shifting, Clearing, and Moving Edges

The *Edit Waveform Edges* dialog operates on the edges of the selected signals. The edges can be removed, shifted in time, or transformed using a equation.

To Edit Waveform Edges:

- 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.
- Select the **Edit > Edit Waveform Edges** menu to open the *Edit Waveform Edges* dialog.

- The **Range** section defines the time range over which to operate.
- **Clear Edges in Range** deletes all the edges but leave a blank waveform.
- **Delete Edges in Range and Shift back edges after range** to deletes all the waveforms and remove that section of time.
- **Shift Edges in Range** to quickly shift the edges by a positive or negative amount of time.



Edit Waveform Edges

Range:
From: -0.001 ns To: 159 ns

Edit:
( ) Clear Edges in Range
( ) Delete Edges in Range, Shift back edges after range
( ) Shift Edges in Range:
Amount of time to shift: 100
(●) Transform Edge Times in Range:
Edge Time Equation: $time + 100

- Choose **Transform Edge Times in Range**, to apply an equation to all the edges. The variable that represents the current time for the edge is $time.

  An example of an edge equation is:

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

  This will take the current time, add ten, and then divide the result by four. Note that this function can also be used to shift edges in a range, but the dedicated shift function is considerably faster.

- Click the **Apply** or **OK** button to modify the waveform edges. Applying will leave the *Edit Waveform Edges* dialog open for other waveform modifications.

1.7 Copy Signals or Waveforms

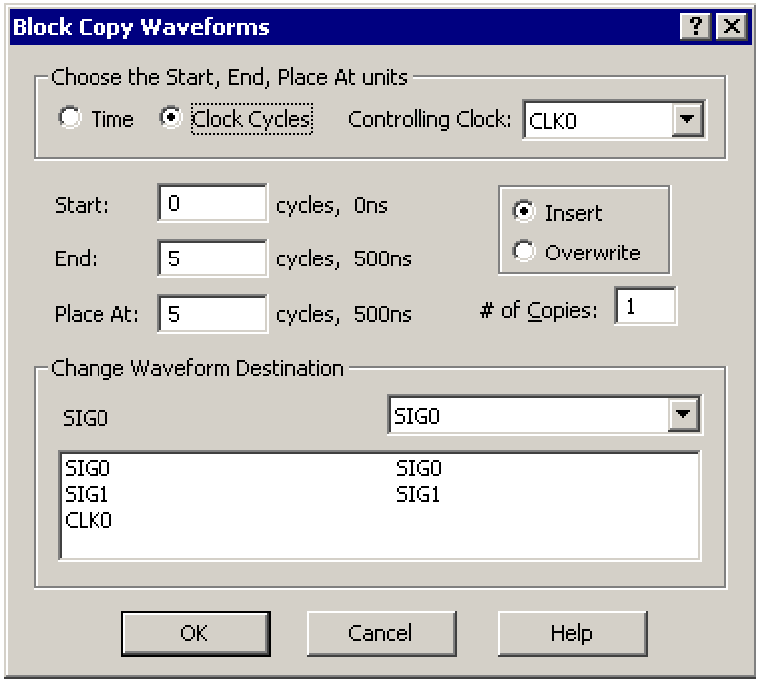
Signals or sections of waveforms can be copied and pasted. The block copying of waveforms can be used to copy whole transactions, like a read cycle, and paste it as needed throughout the diagram.

Copy a signal:

- Select one or more signal names, and then choose the **Edit > Copy Text and Signals** menu. Now the signals and any objects attached to the signals, like text objects and parameters, are copied to the clipboard.
- Paste the signals using the **Edit > Paste** menu option, into either the same timing diagram or into another timing diagram.

Copy a portion of the waveforms (a time slice):

- Select the names of the signals that you want to copy. If no signals are selected, then the copy will operate on all of the signals in the diagram (this is the most common usage).
- Choose the **Edit > Block Copy Waveforms** menu option to open the *Block Copy Waveforms* dialog.


Block Copy Waveforms

Choose the Start, End, Place At units: Time [✓ Clock Cycles] Controlling Clock: CLK0

Start: 0 cycles, 0ns
End: 5 cycles, 500ns
Place At: 5 cycles, 500ns

[✓ Insert] Overwrite

# of Copies: 1

Change Waveform Destination

SIG0 → SIG0

SIG0	SIG0
SIG1	SIG1
CLK0	

- Choose either **Time** or **Clock Cycles** 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 **controlling clock**. If you select **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*.

- The **Start** and **End** define the range of the block to be copied, and the **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 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 select a row and then use the drop-down box to choose a different destination signal.

Short-Cut Method for Waveform Block Copy

- Select the names of the signals that you want to copy. If no signals are selected, then the copy will operate on all of the signals in the diagram (this is the most common usage).
- Hold the **<CTRL>** key while ***clicking and dragging the mouse across the time line*** to select the start and end times of the block. When you release mouse the Block Copy Waveforms dialog will open with the range entered.

Merge two timing diagrams together:

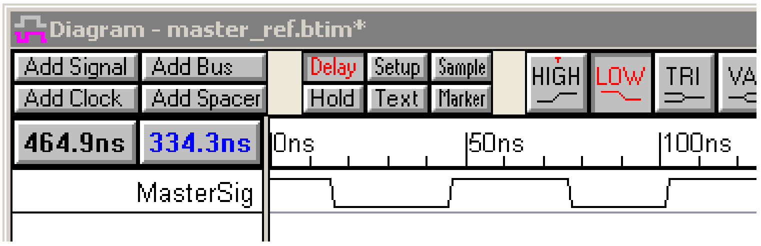
- Open the timing diagram that will hold all of the merged signals, using **File > Open Timing Diagram** menu.
- Then select the **File > Merge Timing Diagram** menu and choose the second timing diagram file. This will copy the entire diagram and paste it into the first diagram.
- See Section 10.4 Merging Diagrams 144 for more information on resolving parameter name conflicts that might occur during a merge.

1.8 Referencing Waveforms from Libraries

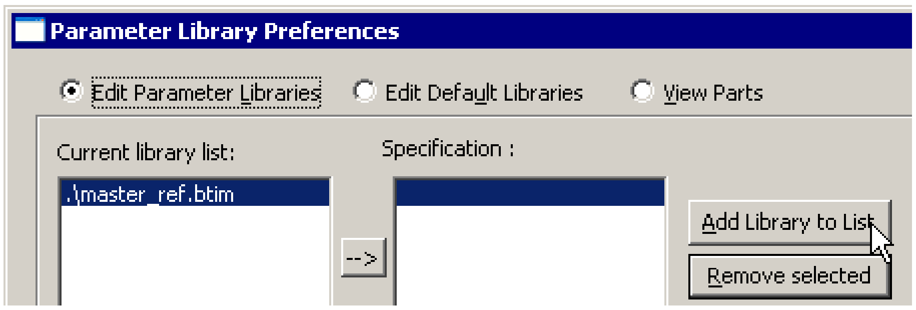
Waveforms can also be referenced from libraries in a similar manner to parameter libraries (see Chapter 10: Parameter Libraries 138). When a master waveform changes, the signals in other diagrams will also change on the next opening of the diagram. This lets you build up libraries of master signals that can be referenced by several different diagrams to show different timing shots of the circuit. The reference waveforms can be contained in external files or from signals in the same file. The Waveform library feature can also be used to display bit-slices of an existing waveform.

Using an External Waveform Library:

- Create a Waveform Library by saving a timing diagram that has the master waveforms in it.

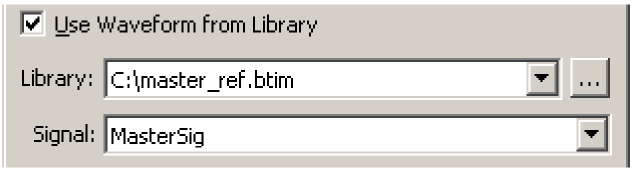


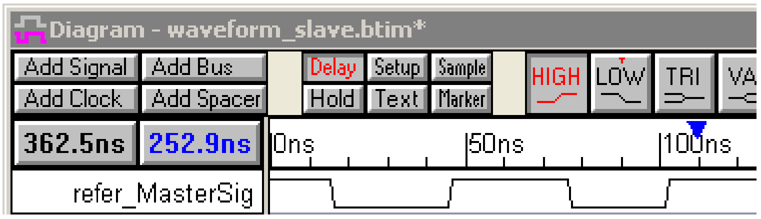
- In the timing diagram that is to use the library, select the **ParameterLibs > Parameter Library Preferences** menu to open the *Parameter Library Preferences* dialog. Then press the **Add Library to List** button and add the Waveform Library to the current library list (Waveform libraries do not need specifications).



Referencing the Library Waveform:

- Double click on a signal, to open the *Signal Properties* dialog, and check the **Use Waveform from Library** box.



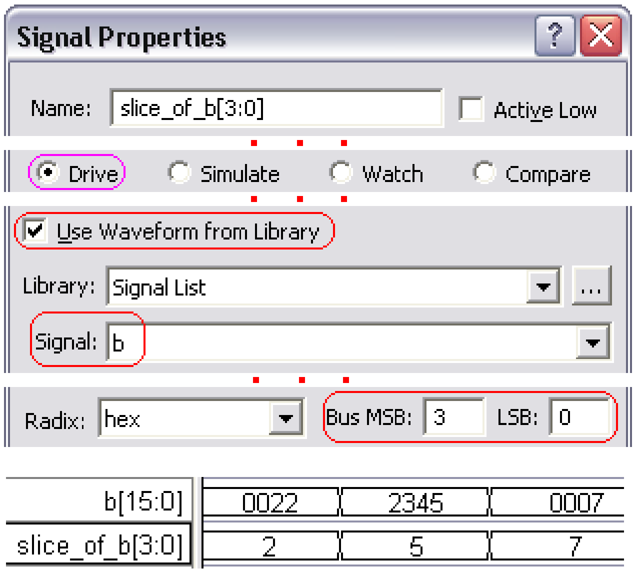


- Use the **Library** control to specify the library that contains the waveform to reference. The **Signal List** library is a list of all waveforms in the current timing diagram.
- Pick a waveform to reference from the **Signal** control.

Making a Bit-Sliced Copy of a Signal

- This method can be used to bit-slice both simulated and non-simulated signals. If you want to bit-slice simulated signals (e.g.signal type **watch)**, then you don't need to create an additional signal: you can just change the MSB and LSB of the original signal or a copy of it as shown in **Section 3.4: Bit Slicing a Watched Signal** of the **BugHunter and VeriLogger** manual.
- Create a new signal by pressing the **Add Signal** button.

- Double click on the new signal to open the *Signal Properties* dialog.
- Check the **Use Waveform from Library** check box. This will display the Library and Signal boxes in that section of the dialog.
- Choose a Library. The **Signal List** library contains signals in the current diagram. Other library choices will also be available if you have added waveform libraries to your diagram.
- Choose a **Signal** from the library. This is the signal to be sliced.
- Set the MSB and LSB to the desired slice. The bits of slice can also be reversed by swapping the MSB/LSB values (e.g. [0:3] instead of [3:0]).


Signal Properties

Name: slice_of_b[3:0] Active Low

[✓ Drive] Simulate Watch Compare

[✓ Use Waveform from Library]

Library: Signal List

Signal: b

Radix: hex Bus MSB: 3 LSB: 0

1.9 Hiding Signals

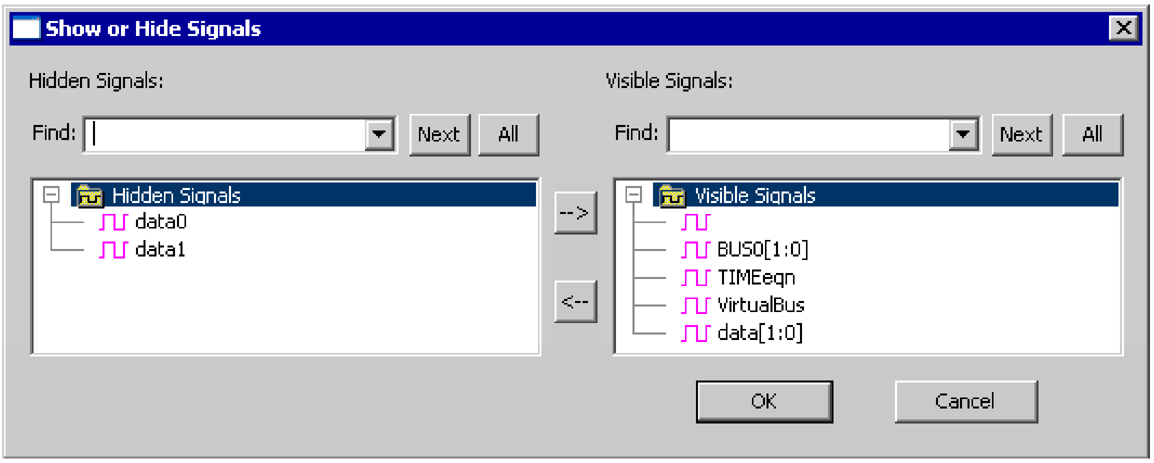
Hiding a signal removes the signal and any attached parameters and text from the screen, but not from the timing diagram itself. Delays, setups, and holds will continue to function, but timing errors will be shown only in the *Parameter* window.

Hide selected signals:

- Select the signal names and then choose **Hide Selected Signals** from the right click context menu or the **View** main menu.
- **View > 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 and Hide using a dialog

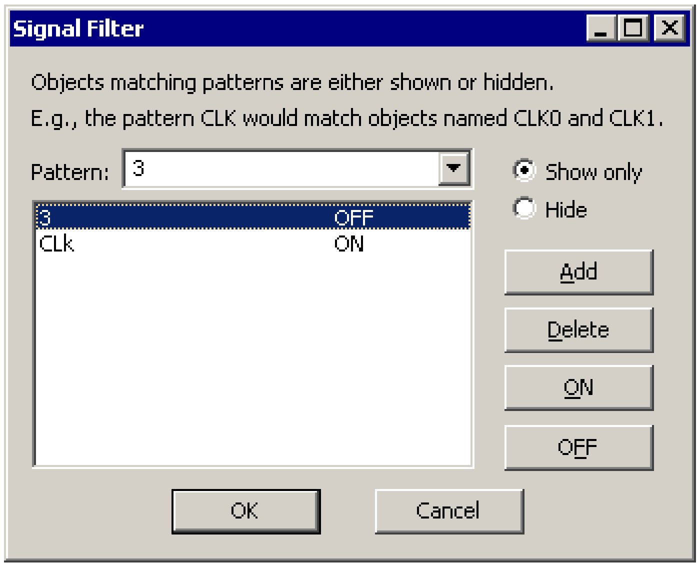
- Choose the **View > Show and Hide Signals** menu to open a *Show or Hide Signals* dialog.



- Move signals to and from the Hidden Signals list on the left and the Visible Signals list on the right using the arrow buttons.
- The **Find** fields, above each list, enable you to select all the signals according to a pattern matching technique. The pattern can be any regular expression, see §15.7: Pattern Matching (Regular Expressions) 193. If you type a pattern (such as 'G1') into one of the fields and click the **All** button, then all signals that contain a match for the pattern (like SIG1, SIG10, or SIG11) will be selected. If you type a pattern and click **Next** button a few times, then it will successively select the next few signals that match the pattern.

Hide using a filter pattern:

- Choose the **View > Filter Signals** menu 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.


Signal Filter

Objects matching patterns are either shown or hidden.
E.g., the pattern CLK would match objects named CLK0 and CLK1.

Pattern: 3

3	OFF
CLk	ON

[✓ Show only] Hide

- 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. The **On** and **Off** buttons can be used to selectively activate the filters. These patterns control what signals will be displayed and will also determine if newly added signals will be displayed, depending on whether the new signals match against a filter pattern.

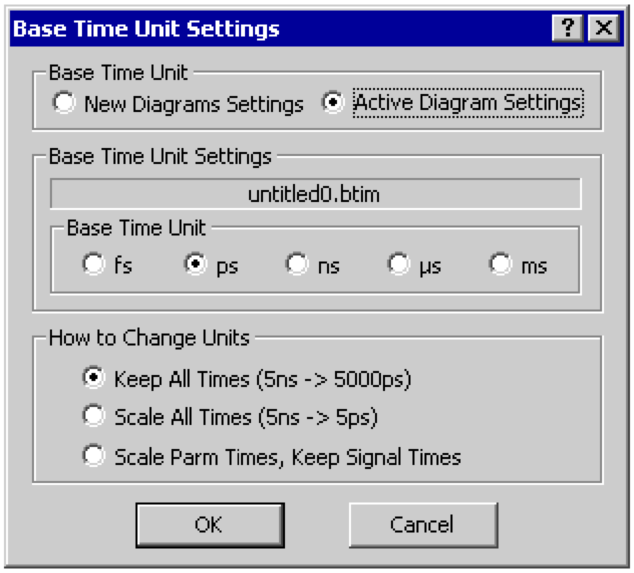
1.10 Base and Display Time Units

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.

The program has a limit of approximately 1 billion time units (varying with screen resolution and size). For this reason the base time unit should be chosen to match the range of times needed by your design.

To set the Base Time Unit:

- Select the **Options > Base Time Unit** menu to open the *Base Time Unit* dialog.
- 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.


Base Time Unit Settings

Base Time Unit: New Diagrams Settings [✓ Active Diagram Settings]

Base Time Unit Settings: untitled0.btim

Base Time Unit: fs [✓ ps] ns µs ms

How to Change Units: [✓ Keep All Times (5ns -> 5000ps)] Scale All Times (5ns -> 5ps) Scale Parm Times, Keep Signal Times

To set the Display Time Unit:

- Choose the **Options > Display Unit** menu item to display a submenu of display time units, and click on a time unit to make it the new Display Time Unit.

1.11 Selecting, Moving, and Sorting Signals

Signals can be selected, moved, or sorted using any of the techniques below:

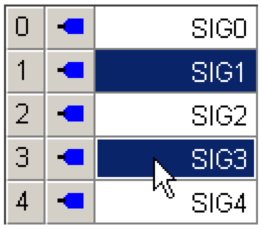
To select a signal

• Select **one signal**: left mouse click on the signal name.	**Left Mouse Click**
• Select **nonconsecutive signals**: left mouse click while holding the CTRL key down.	**<CTRL> + Left Mouse**
• Select **consecutive signals**: left mouse click	**click to start, then <SHIFT> + <UP>**

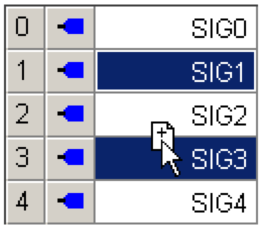
to pick the first one, then hold the SHIFT key while pressing the Up or Down arrow keys.	**or <SHIFT>+ <Down>**
• Select **consecutive signals**: left mouse click and drag.	**Left Mouse click and drag**
• Select **consecutive signals**: hold the SHIFT key down while left mouse clicking on signals labels.	**<SHIFT> + Left Mouse**
• Select **all signals** in a diagram by pressing CTRL-A.	**<CTRL> + <A>**
• **Unselect** by using the ESC key or by clicking outside the signal label window.	**ESC**

To move a signal:

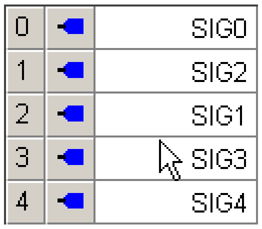
- Select the signal names ***in the order that the signals should be arranged after the move*** by either:
  - Left mouse click on one signal name to select it.
  - Hold the CTRL and select several signal names.
  - Left click and drag through several names to select several signals that are contiguous in the label window, OR hold the SHIFT key and drag.



- Release the mouse if you have been holding it down.
- Click down on a selected signal so a paper icon appears, and then drag the icon it to the new location to move the signals.

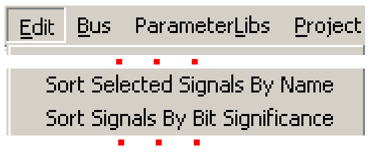


- Drop the paper icon by releasing the left mouse button. The signals will be inserted just below the green bar, ***in the order in which they were selected***.



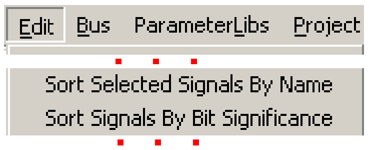
Sort Signal Names Alphabetically

- Sort selected signals alphabetically name by choosing the **Edit > Sort *(Selected)* Signals By Name** menu. If no signals are selected then it will rearrange all of signals in the diagram. If some signals are selected then it will only rearrange the selected signals.

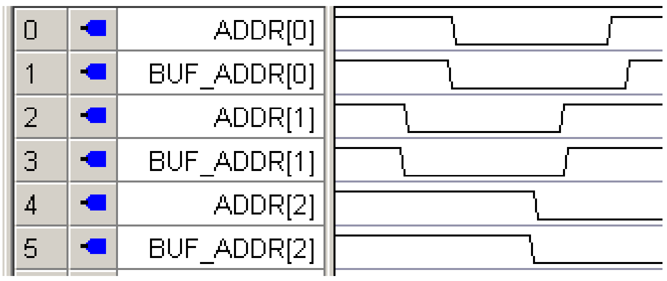

Edit
Bus
ParameterLibs
Project
Sort Selected Signals By Name
Sort Signals By Bit Significance

Sort Signals By Bit Significance:

- Choose the **Edit > Sort Signals by Bit Significance** menu to reorder the signals.

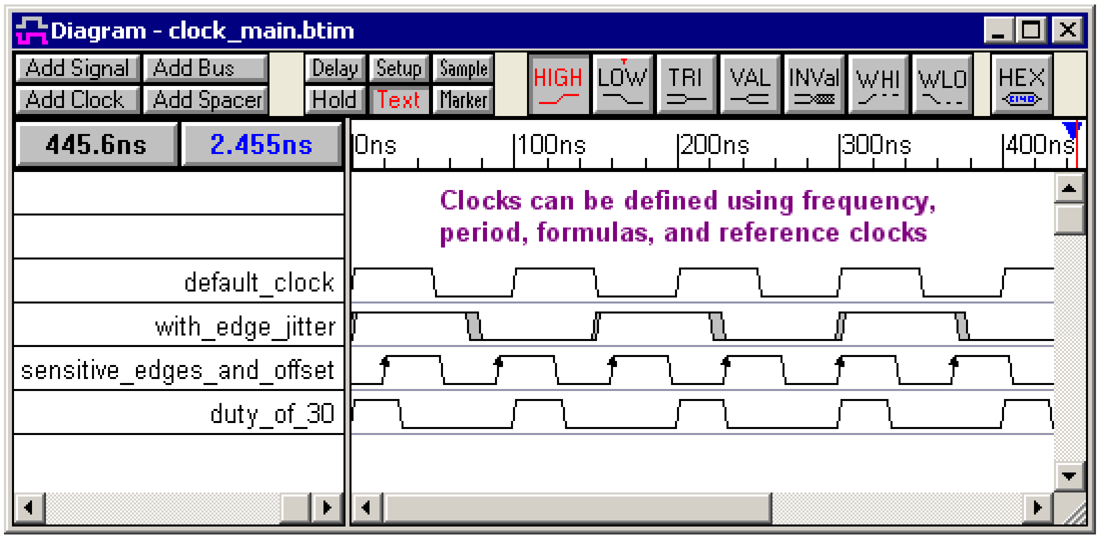

Edit
Bus
ParameterLibs
Project
Sort Selected Signals By Name
Sort Signals By Bit Significance

- Notice in this example, the individual bits of two group busses are interleaved.



Chapter 2: Clocks and Time Formulas

Clocks are special repetitive signals that are automatically drawn based on their attributes: period, frequency, duty cycle, edge jitter, offset, and other parameters. Clocks can be related to other clocks by using the Reference Clock property or by using time formulas that reference another clock's attributes like period, offset, and jitter.

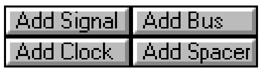


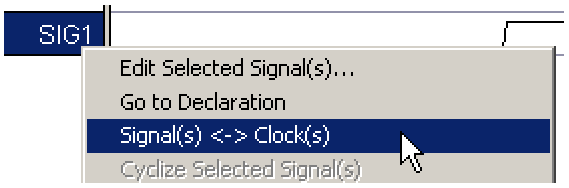
2.1 Adding Clocks

Clocks can be created using the **Add Clock** button or converted from a signal through the right click menu. Clocks can be 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.

Add a new clock:

- Press the **Add Clock** button, to create a new clock and open the *Edit Clock Properties* dialog.
- Or, if ***importing from a VCD or logic analyzer*** the clock data will import into regular signal and will need to be converted into a clock. Right click on a signal name and choose **Signal(s) <-> Clock(s)** option from the menu. The editor will attempt to determine the parameters of the clock (period, offset, duty cycle, etc.) from the signal waveform; if editor cannot determine these parameters, the clock will have the default parameters.





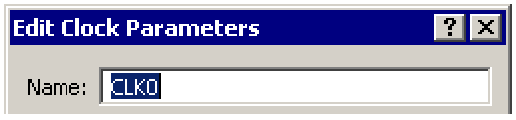
Open the Edit Clock Parameters dialog:

Most of a clock's parameters are edited using the *Edit Clock Parameters* dialog. This is a fully interactive dialog that changes the clock as soon as you change a clock attribute, so you can

Copyright © 2011, SynaptiCAD Sales, Inc.

immediately see the impact on your design.

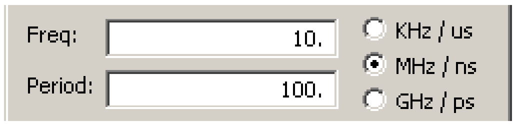
- Double-click the waveform of the clock to open the *Edit Clock Parameters* dialog.
- Or, double click on the clock name and press the **Clock Properties** button.


Edit Clock Parameters
Name: CLK0

Define Clock Period or Frequency (three ways):

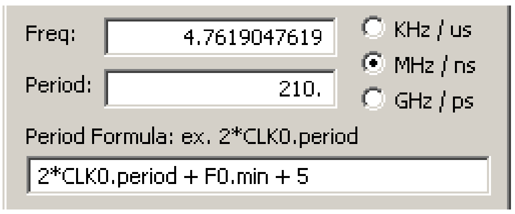
1) By fixed time or frequency:

- 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, lower the base time unit (see 1.10 Base and Display Time Units 36).


Freq: 10.
Period: 100.
( ) KHz / us
(●) MHz / ns
( ) GHz / ps

2) By a time formula based on other clocks and parameters. This is particularly useful for describing circuits in which the clock frequency and delay values change depending on the speed grade of the part being used or for phase-locked loop (PLL) *frequency multiplier circuits*. By storing a clock frequency in a parameter, and referencing that parameter in the frequency box of the clock, the entire diagram can be changed by loading in a different parameter library for each speed grade.

- When a **Period formula** is entered, the **period** and **frequency** boxes will update to show the calculated times. See Section 2.5: Time Formulas 47 for all of the rules for formulas.


Freq: 4.7619047619
Period: 210.
( ) KHz / us
(●) MHz / ns
( ) GHz / ps
Period Formula: ex. 2*CLK0.period
2*CLK0.period + F0.min + 5

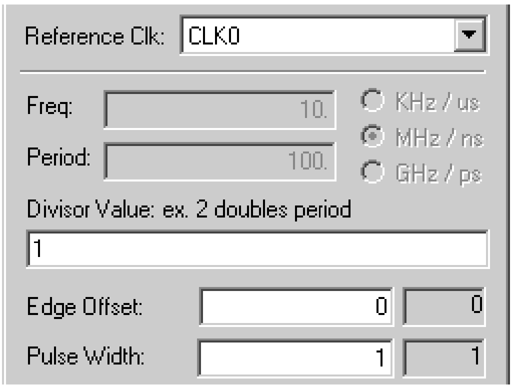
- *Clocks* can reference another clock's **period**, **duty**, **offset**, and rising **(jrise)** and falling **(jfall)** edge jitter, and the four buffer delays (**minLH**, **maxLH**, **minHL**, **maxHL**).

```
clockname.period
clockname.duty
clockname.minLH
```

3) By a Reference Clock which tightly relates two clocks and is the most accurate way to model a *clock divider* implemented with digital logic. This also correctly models *delay correlation* effects between the reference clock and the sub-clock.

- When you choose a **reference clock**, the period, offset, and duty cycle edit boxes are replaced by three new edit boxes called **Divisor**, **Edge Offset**, and **Pulse Width** which are all tightly related to the reference clock.
- The **Divisor** affects the period of the new clock. A divisor of 2 makes the new clock have twice the period of the reference clock (half the frequency, acts as a clock divider

circuit). A divisor of 0.5 divides the period by 2 (acts a frequency multiplier circuit, e.g. phase-locked loop circuit).



Reference Clk: CLK0

Freq: 10.
Period: 100.
( ) KHz / us [✓ MHz / ns] ( ) GHz / ps

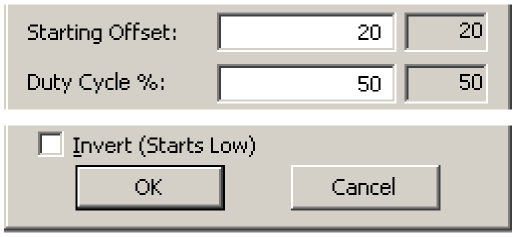
Divisor Value: ex. 2 doubles period
1

Edge Offset: 0 0
Pulse Width: 1 1

- **Edge Offset** is the number of edges of the reference clock that the new clock waits until starting. A value of 0 starts the new clock on the rising edge of the reference clock. A value of 1 starts the new clock on the first falling edge of the reference clock. (Assuming the reference clock is not inverted). This only has meaning when frequency is being divided; multiplied clocks ignore this value. This box can accept a time formula.
- **Pulse Width** is the number of edges of the reference clock that determine the width of the new clock. To get a 50% duty cycle the Pulse Width must equal the Divisor. For example a clock with a divisor of 2, has a period equal to 2 periods of the reference clock. If you count the edges on the reference clock (0,1,2) will bring you to the beginning of the second period of the reference clock. So a pulse width of 2 will give you a 50% duty cycle. This only has meaning when frequency is being divided, multiplied clocks ignore this value.

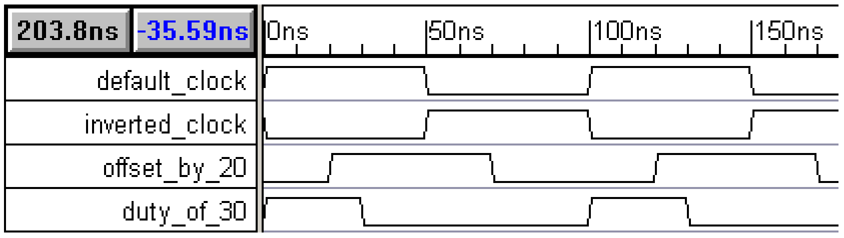
Offset, Duty Cycle, and Invert

- The first default clock transition is a rising edge at time zero. The **offset** value shifts the clock transitions by an amount of time.This can be a formula.
- **Duty cycle** is the percentage time that a clock is high during a given period. This can be a formula.



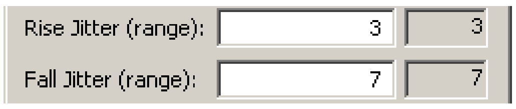
Starting Offset: 20 20
Duty Cycle %: 50 50
[ ] Invert (Starts Low)

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

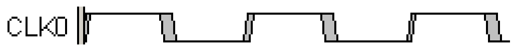


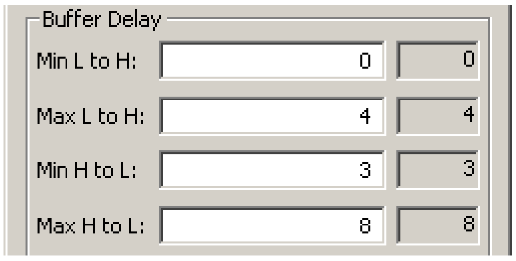
Edge Jitter and Buffer Delays

- Edge Jitter adds uncertainty ***around the clock edge***, and is used to model PLL jitter or frequency fluctuations in a crystal oscillator.
- Buffer delays add uncertainty ***after the calculated edge*** time, and models the delay of a clock passing through a buffer. Clock skew can be model using the buffer delays and the delay correlation boxes (see Section 2.4: Modeling Clock Skew 46).
- The Jitter and Buffer boxes can also accept time formulas.



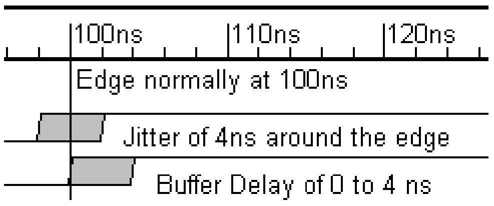
Rise Jitter (range):	3	3
Fall Jitter (range):	7	7





Buffer Delay

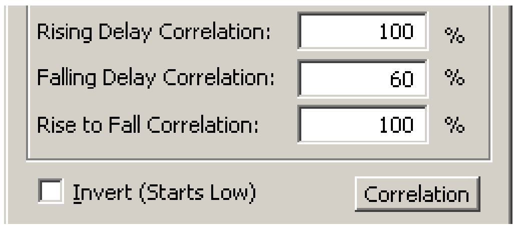
Min L to H:	0	0
Max L to H:	4	4
Min H to L:	3	3
Max H to L:	8	8



Buffer Delays for clocks are automatically correlated:

Normally, buffer delays are 100% correlated, because the uncertainty from the buffer doesn't add "edge-to-edge" uncertainty to the clock's edges. In other words, the same amount of uncertainty will exist between all the edges of the clock, so it can be ignored for "edge-to-edge" related measurements and for delay chains that start on the clock's edges.

- Although rarely necessary, you can modify the delay correlations directly in the *Edit Clock Parameters* dialog.



Rising Delay Correlation:	100	%
Falling Delay Correlation:	60	%
Rise to Fall Correlation:	100	%

☐ Invert (Starts Low)

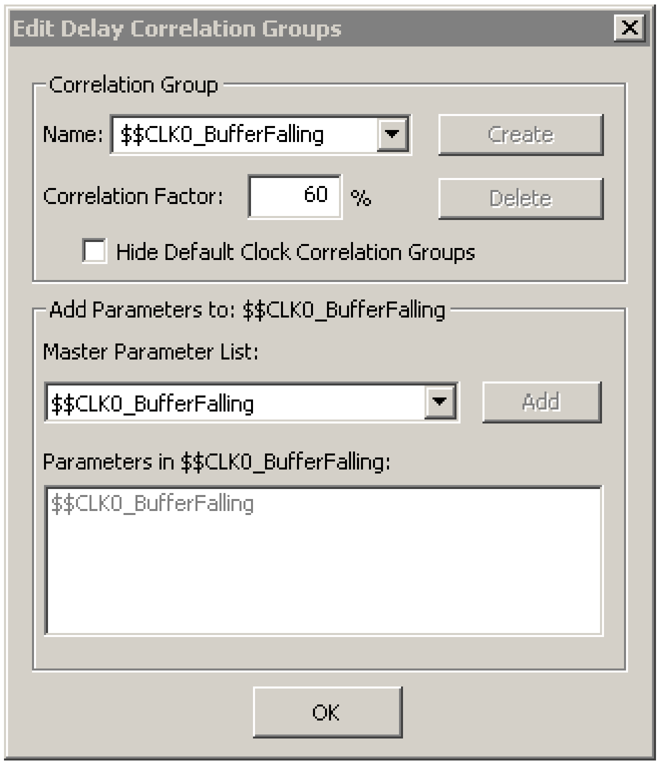
- They can also be edited by pressing the **Correlation** button to open the *Edit Delay Correlation Groups* dialog.
- Uncheck the **Hide Default Clock Correlation Groups** box.
- Each clock will have three default correlation groups:

$$*cname*_BufferRising

$$cname_BufferFalling

$$cname_BufferRisingFalling

- Each default group will have one delay by the same name. In the diagram window, each buffered clock edge is affected by the corresponding delay.


Edit Delay Correlation Groups

Correlation Group

Name: $$CLK0_BufferFalling

Correlation Factor: 60 %

☐ Hide Default Clock Correlation Groups

Add Parameters to: $$CLK0_BufferFalling

Master Parameter List: $$CLK0_BufferFalling

Parameters in $$CLK0_BufferFalling: $$CLK0_BufferFalling

Display tricks for Clocks:

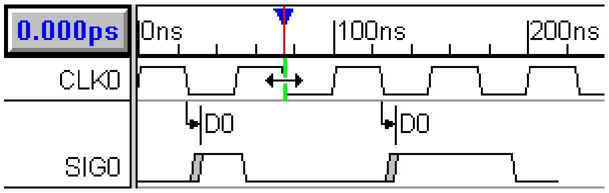
- To make just a clock have straight edges, while all of the rest of the signals have slanted edges, double click on the clock name to open the *Signal Properties* dialog, the press the **Analog Props** button to open the *Analog Properties* dialog, then check the **use straight edges** control. To make all of the signals have straight edges see Section 1.4 Display Settings for Signals 22.
- To add virtual state information to segments on a Clock, press the **Hex** button and then double-click on the middle of the segment to open the *Edit Bust State* dialog. If the Hex button is not pressed, the double-click will open a different dialog to allow editing of the clock.
- To add little arrows on the edges of clocks, double click on the clock name to open the *Signal Properties* dialog and check the **Falling Edge Sensitive** or **Rising Edge Sensitive** boxes.

2.2 Inserting and Deleting Clock Cycles

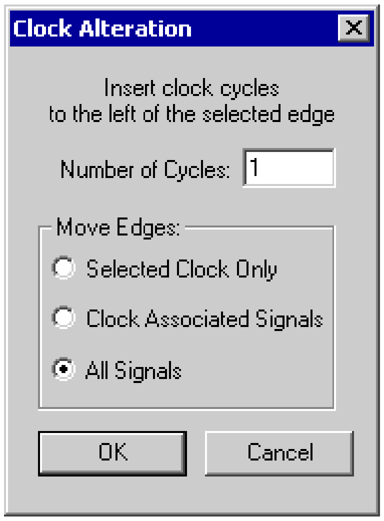
Clock cycles and be inserted or deleted, so that parameters attached to a particular clock edge will move in time.

To insert or delete cycles in a clock:

- Select a clock edge for the insertion point, or the first edge of the clock cycle to be deleted



- To Insert, a press the **<Insert>** key or choose the **Edit > Insert Clock Cycles** menu.
- To Delete, press the **<delete>** key or choose the **Edit > Delete Clock Cycles** menu.
- These actions will open the *Clock Alteration* dialog in either insert or delete mode.
- Enter the **number of cycles** to insert or delete.



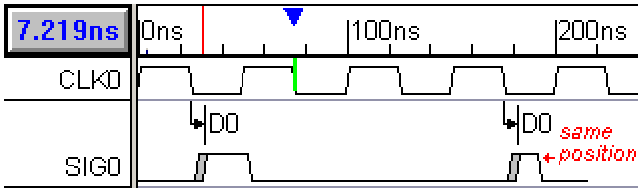
Clock Alteration

Insert clock cycles
to the left of the selected edge

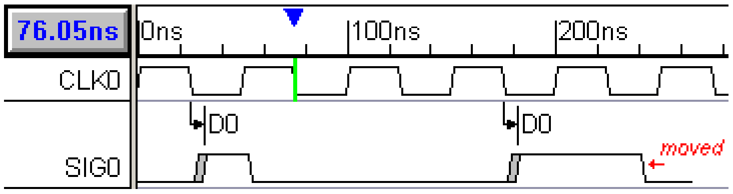
Number of Cycles: 1

Move Edges:
( ) Selected Clock Only
( ) Clock Associated Signals
(●) All Signals

- The **Selected Clock Only** radio button causes only the edges of the selected clock to move.
- The **Clock Associated Signals** radio button causes the edges of all signals using the selected clock as their clocking signal to move



- The **All Signals** radio button causes the edges of all the signals in the diagram to move.

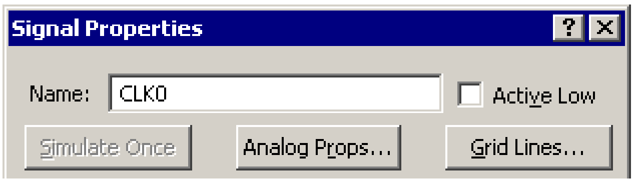


2.3 Grid Lines on Clocks and Signals

Grid lines are vertical lines drawn from the edges of a clock or on any signal.

To draw a grid lines:

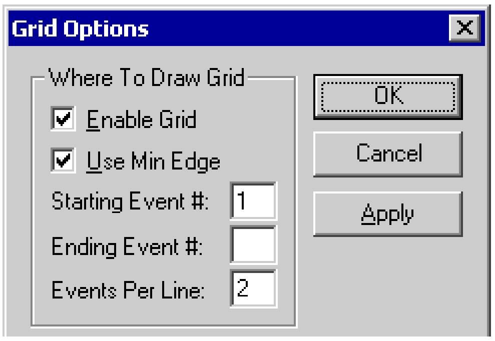
- Double-click on the clock name to open the *Signal Properties* dialog, and press the **Grid Lines** button in the top right of the dialog. This opens the *Grid Options* dialog.



Signal Properties

Name: CLK0 [ ] Active Low

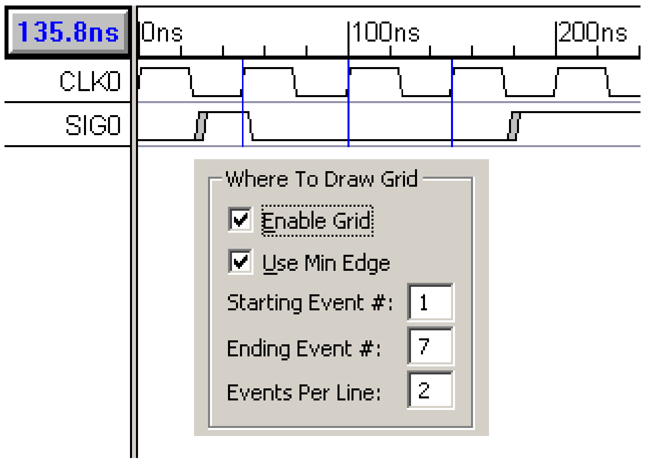
- Check the **Enable Grid** checkbox. This enables the rest of the grid options. If you later decide to turn off the grid then uncheck this check box.
- Fill in the rest of the controls to design a grid pattern. Use the **Apply** button to test your grid settings.



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

Where to Draw Grid Section:

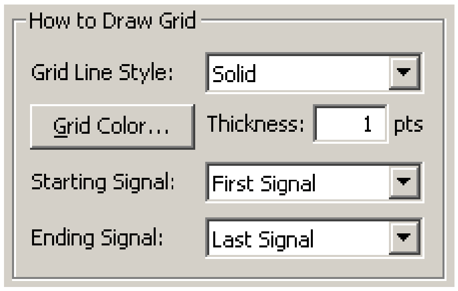
- The **Use min Edge** check box determines if grid lines are drawn from the minimum or maximum edge of a clock transition.
- The **Starting Event #** is the event number where the first grid line is to be drawn. The first event in the clock is the 0 event.
- The **Ending Event #** is the event number where the last grid line is to be drawn. If blank, the will draw on all the clock edges.
- The **Events per Line** edit box determines how many clock transitions occur before another grid line is drawn. A "1" draws a grid line on every transition, and "2" skips one transition between grid lines, etc.



Enable Grid ✓
Use Min Edge ✓
Starting Event #: 1
Ending Event #: 7
Events Per Line: 2

How to Draw Grid Section:

- The **Grid Line Style** 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 and **Thickness** sets the width of the line.
- The **Starting Signal** and **Ending Signal** determine the length of the grid lines. The default values cause the grid lines to be drawn from the top of the diagram to the bottom.



Grid Line Style: Solid
Thickness: 1 pts
Starting Signal: First Signal
Ending Signal: Last Signal

Hidden Signals and Text Objects:

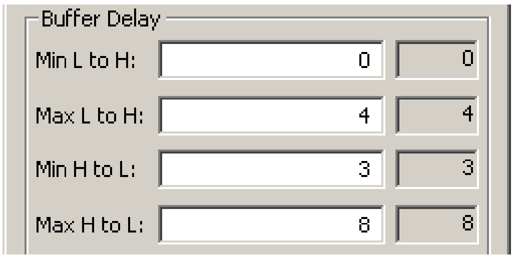
Normally, hiding a signal will also hide the grid lines and text attached to the signal. However, sometimes it may convenient to use a signal just for holding grid lines (or text), but otherwise that signal is unrelated to your circuit design and does not need to be displayed. The checking **View > Show Hidden Text** menu, then hiding the documentation signal will allow the grid lines to continue to be displayed even though the signal is hidden.

2.4 Modeling Clock Skew with Delay Correlation

Datasheets for clock tree buffer ICs often include a timing skew parameter that can be used to compute the delay correlation between the buffer gates inside the IC. To take advantage of the skew information, you must first specify a buffer gate delay for each of the clocks being buffered by the clock tree. Next, you will compute a delay correlation from the skew value. Finally, you can create a correlation group with the calculated delay correlation value that contains the clock buffer delays. This delay correlation group will automatically account for the clock skew in all the timing calculations related to the clocks.

1) Enter the buffer delays into each buffered clock

- For each clock, double click on the waveform to open the *Edit Clock Parameters* dialog and enter the buffer delays in the **Buffer Delay** section of the dialog. Close the dialog when you are done with all the clocks.



Buffer Delay

Min L to H:	0	0
Max L to H:	4	4
Min H to L:	3	3
Max H to L:	8	8

This causes the program to automatically generate two delay parameters (one for the rising edge delay and one for the falling edge delay) for each clock. These are the parameters that you will add to the correlation group in step 3.

2) Compute the delay correlation using the skew value for the clock tree buffer:

- First, use the min and max buffer delays from your data sheets to calculate the buffer uncertainty.

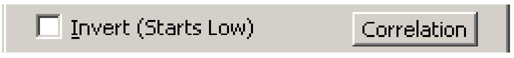
  BufferUncertainty = MaxBufferDelay - MinBufferDelay

- Then use the skew value from the data sheet to calculate the delay correlation.

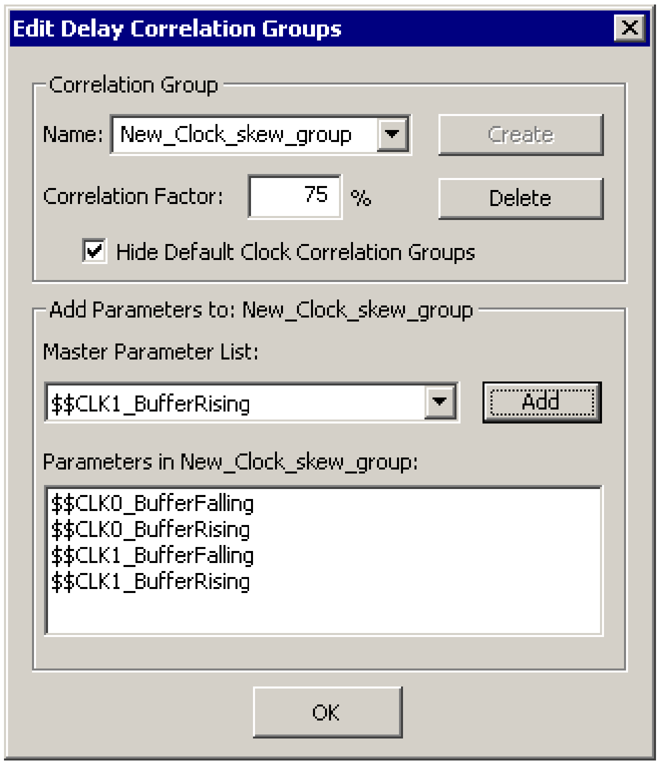
  DelayCorrelation% = (1 - (Skew / BufferUncertainty)) * 100

3) Create the Correlation group for the clock buffer delays

- Double click on one of the clock waveforms to open the *Edit Clock Parameters* dialog and press the **Correlation** button to open the *Edit Delay Correlation Groups* dialog.



- To create a new group, type in a name and press the **Create** button.
- Enter the **DelayCorrelation%** value into the **Correlation** edit box.
- Add the two buffer delays for each buffered clock to the correlation group by selecting them from **Master Parameter List** and pressing the **Add** button. Each clock will have two delays named **$$***ClockName*_BufferRising and **$$***ClockName*_BufferFalling.
- Note that the finished correlation group should have at least 4 delays (2 for each clock being correlated). There will be two more delays for each additional buffered clock.



Name: New_Clock_skew_group
Correlation Factor: 75 %
[✓] Hide Default Clock Correlation Groups
Add Parameters to: New_Clock_skew_group
Master Parameter List: $$CLK1_BufferRising
Parameters in New_Clock_skew_group:
$$CLK0_BufferFalling
$$CLK0_BufferRising
$$CLK1_BufferFalling
$$CLK1_BufferRising

Example of how to calculate delay correlation using skew

Assume you have a datasheet for **three clock buffers** where the buffers have a **2-6 ns propagation delay**, with a **skew of 1 ns** between the buffers. Enter the 2-6 ns delay into the Buffer Delay section of the Edit Clock Parameters dialog of all three clocks being buffered. Next, computing the delay correlation for the buffers, we get:

- BufferUncertainty = 6ns - 2ns = 4ns
- DelayCorrelation% = (1 - (1ns / 4ns)) * 100 = 75%

Create a correlation group with a 75% correlation factor and add the buffer delays to the group (3 clocks x 2 buffer delays per clock = 6 delays to add to group). If the clock buffers were perfectly correlated (100% correlation), the skew would be 0ns. If the clock buffers were not correlated at all (0% correlation), the skew would be 4ns. This large skew (no correlation) is the default timing value that is used when no correlation groups are used.

Side Note: For each clock, two correlation groups are also created which automatically correlate the buffer delays for that clock signal. These groups share the same name as the clock buffer delays (e.g. if the clock buffer delay is called $$CLK0_BufferRising, a correlation group called $$CLK0_BufferRising is also created). These special correlation groups allow the software to automatically ignore uncertainty between the edges of a specific clock that is caused by the clock's buffering. Generally these special correlation groups can be left "as-is".

2.5 Time Formulas for Clocks and Parameters

A Time formula can be used wherever a time value is required. For example the min/max for parameters, clock period, clock offset, and clock jitter all can accept time formulas. Time formulas are in display time units (see 1.10 Base and Display Time Units 36). Some examples of time formulas:

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

- A quick way to grab a parameter name for use in a formula is to click into the min or max box in the *Properties* dialog, then press the **Library** button to open the *View Parameters in Libraries* dialog (see section 10.2 Referencing Parameters 140). This will list all parameters in the project and the associated libraries, once you select a parameter it will be inserted into the box in the *Properties* dialog that you were in.

Time formulas are algebraic combinations of the following:

- **Parameter names, parameters with dot attributes,** and **clocks with dot attributes** can be used in formulas. If just the parameter name is used, 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.

```
D0
D1.min
D2.max
Clk0.period
```

- **Function Calls** defined in the Parameter window. The dot-max and the dot-min properties can be used to force the equation to use a particular value. Also, the input arguments can be used in the expressions.

```
f(4,5)
func.max(2+D0,4)
```

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

```
5
2.010
255E-3
```

- **Constants** are fixed point numbers that are *unit-less*. 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.

```
'2
'4.1e2
```

- **Macro names** enclosed with percentage marks will be replace with the macro value as defined in the *Edit Formula Macro* dialog. See Section 10.1 Adding Libraries, Specifications, and Macros 138 .

```
%lib_spec$.F0
```

Dot Attributes

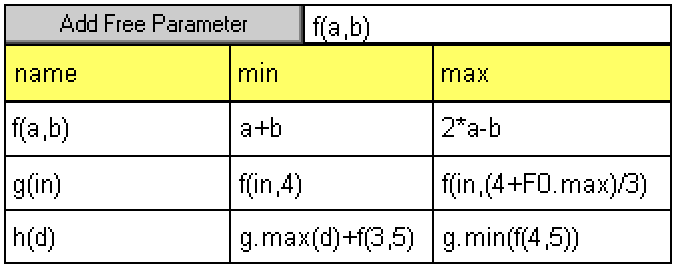
- *Parameters* and *User defined functions* support **min** and **max** attributes.

```
F(5).min
D0.max
```

- *Clocks* can only be referenced with an attribute of **period**, **duty**, **offset**, and **jrise** (rising jitter) and **jfall** (falling jitter), and the four buffer delays: **minLH**, **maxLH**, **minHL**, **maxHL**.

```
Clk.duty
Clk.offset
Clk.jrise
```

Function Definitions in the Parameter Window



name	min	max
f(a,b)	a+b	2*a-b
g(in)	f(in,4)	f(in,(4+F0.max)/3)
h(d)	g.max(d)+f(3,5)	g.min(f(4,5))

- Click the **Add Free Parameter** button in the *Parameter* window, to add a free parameter.
- Double click on the free parameter to open the *Properties* dialog.
- The function name defines both the name and the input argument list. There must be a least one argument. Arguments are separated by commas.

```
FunctionName(input1<,Input2,….,inputN>)
```

- The min and max fields define the body of the function.

Built in Functions supported:

Function	**Description**
abs	Returns the absolute value of the number.
acos	Returns the arccosine of a number in radians. The number must be a value from –1 to 1.
asin	Returns the arcsine of a number in radians. The number must be a value from –1 to 1.
atan	Returns the arctangent of a number in radians.
cos	Returns the cosine of a number expressed in radians.
cosh	Returns the hyperbolic cosine of the number.
degrees	Converts radians to degrees.
exp	Returns e raised to the power of the number.
fact	Returns the factorial of the number. The number must be in the range of 0 to 69. If the number is not an integer, the decimal will be truncated.
int	Rounds the number down to the nearest integer. `Example: int 4.1 = 4`
ln	Returns the natural logarithm (log base e) of the number. The number must be greater than zero.
log	Returns the logarithm to the base 10 of the number. The number must be greater than zero.
radians	Converts degrees to radians.
rand	Returns a random integer between 0 and 32,767 based on the seed number given.
sign	Returns -1 if the number is negative, 1 if the number is positive, and zero if the

	number is zero.
sin	Returns the sine of the number expressed in radians.
sinh	Returns the hyperbolic sine of the number.
sqr	Returns the square root of the number.
sqrt	Returns the square root of the number.
tan	Returns the tangent of the number expressed in radians.
tanh	Returns the hyperbolic tangent of the number.
trunc	Truncates the number to an integer by removing the decimal. `Example: trunc 4.1 = 4`

Mathematical Operators

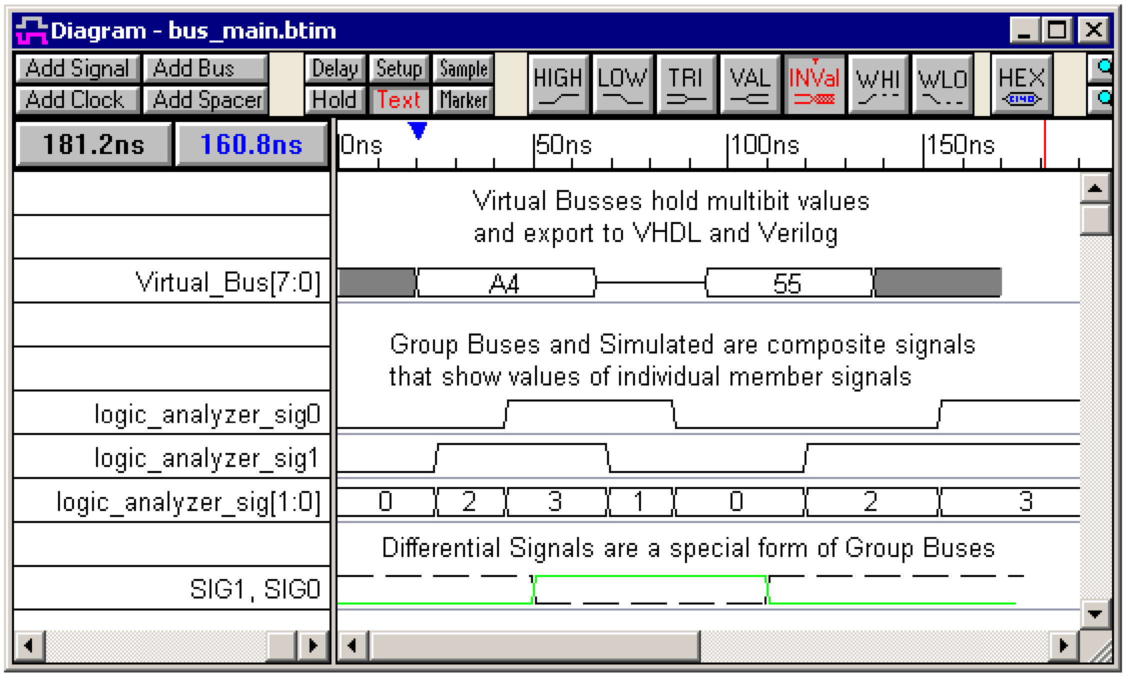
The timing diagram editor supports the **+**, **-**, *****, and **/** mathematical operators. The editor also uses the operator ^ to raise a number to a power.

No Circular Dependencies:

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.

Chapter 3: Buses and Differential Signals

A bus is a multi-bit signal. The timing diagram editor supports three types of buses plus differential signal display:



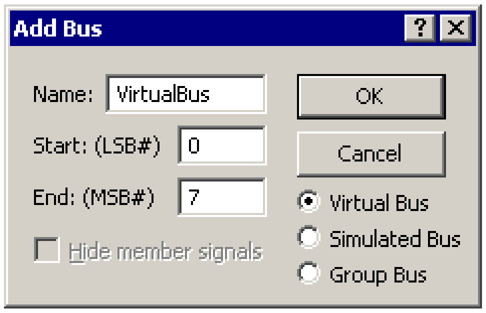
- **Virtual Bus** is a single signal defined as multiple bits. This is the most common and easiest to work with because all of the normal signal editing techniques work on it.
- **Group Bus** displays the aggregate values of its member signals. This is handy way to manage lots of single bit signals that have been imported from other sources.
- **Simulated Bus** is a simulated signal defined as a concatenation of it member signals. This is primarily designed for the testbench products so that both a member signal and the whole bus can be passed to models as needed.
- **Differential Signals** are two-bit group buses that display a superimposed image of the member signal waveforms.

3.1 Virtual Buses

A Virtual bus is a signal that is defined as having multiple bits and generally drawn using valid, invalid, and tristate waveforms. A Virtual bus will export to VHDL and Verilog as a multi-bit signal. This is the most common type of bus and is the easiest to draw and work with.

Create a virtual bus:

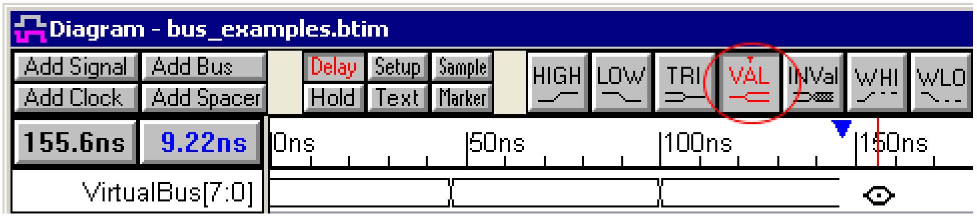
- **Fastest method:** Make sure no signals are selected, then click the **Add Bus** button to open the *Add Bus* dialog. Then select the **Virtual Bus** radio and set the **MSB** and **LSB** values.
- **Alternate method:** Add a signal and then double-click on the name to open the *Signal Properties* dialog. In the dialog edit the edit the **MSB** and **LSB** values.



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

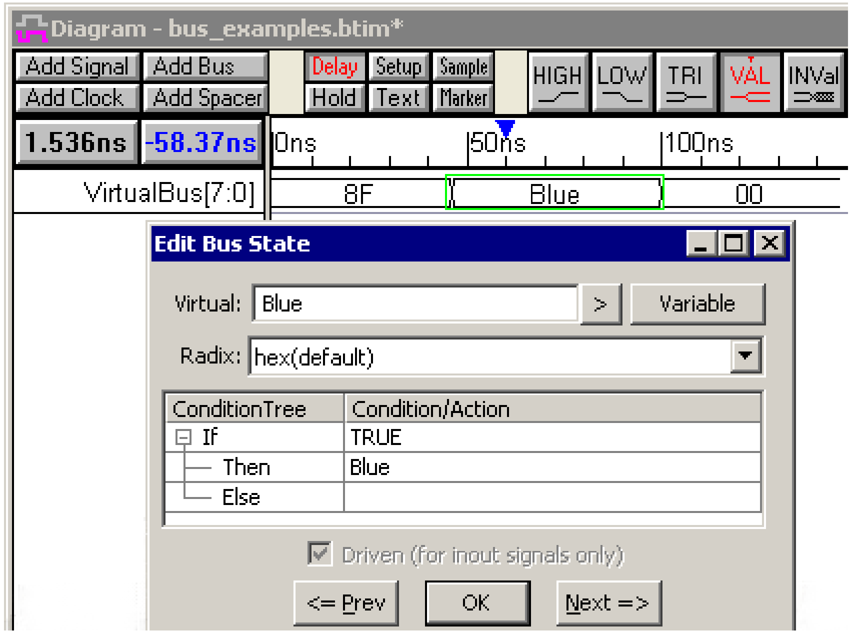
Draw Virtual Bus Waveform Segments:

- Sketch the bus waveform using any of the waveform states. To quickly draw consecutive valid states click twice on the **Valid** state button (or double click on it) so that it is red and also has the red T on the top of the button. This will stop the button from toggling.



Define the Values of the Waveform Segments:

- Open the *Edit Bus State* dialog by either double-clicking on a segment OR first selecting a segment and then clicking the **HEX** button on the button bar.



Virtual: Blue
Radix: hex(default)

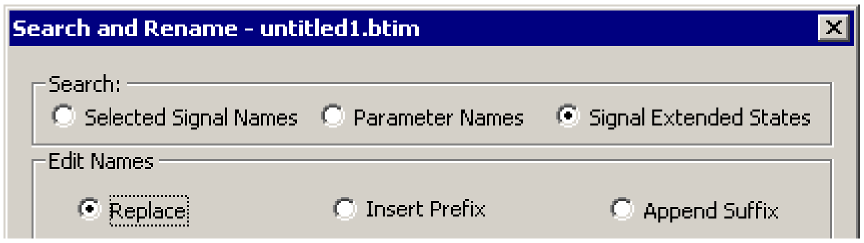
ConditionTree	Condition/Action
If	TRUE
Then	Blue
Else	

[✓] Driven (for inout signals only)

- In the **Virtual** field, type in the segment value. This can by any type of data including text with spaces (e.g., A0C, 5 + 3, blue level, and 24 are all valid virtual states).
- Use **Next** and **Prev** buttons, or the **<Alt>-N** and **<Alt>-P** keys, to move between the different segments on the same bus.

Search and Rename Bus State Values

- Choose the **Edit > Search and Rename** menu to open a dialog.


Search and Rename - untitled1.btim

Search: ( ) Selected Signal Names ( ) Parameter Names [✓ Signal Extended States]

Edit Names: [✓ Replace] ( ) Insert Prefix ( ) Append Suffix

- Select **Signal Extended States** and the fill the rest of the fields with regular expressions that will match to a state value. See section 11.1 Signal Names for export and import 149 for examples of the regular expressions that can be used in the dialog.

Bit-Slicing Virtual Buses

- To display a bit-slice of a virtual bus, use the Waveform Library feature described in Section 1.8 Referencing Waveforms from Libraries 32.
- To display a bit-slice of a simulated (watched) signal, you don't need to use the waveform library feature (although it will also work), just change the MSB/LSB of the signal or a copy of the signal as shown in **Section 3.4: Bit Slicing a Watched Signal** of the **BugHunter and VeriLogger** manual.

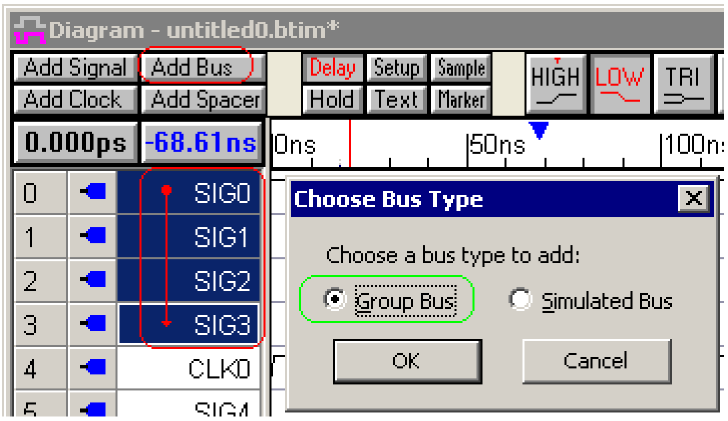
3.2 Group Buses

Group buses are composite signals whose transitions and state values are determined by their member signals. Group buses are usually created from single bit signals that have been imported from logic analyzers or VCD files so that the aggregate value of the bus can be easily viewed. A group bus can either be edited directly by changing the bus value or indirectly by editing a member signal. Member signals can be hidden or displayed as needed. During most exporting operations (e.g. to VHDL or Verilog), only the member signals will be exported. The group bus will not be exported since it only duplicates the information in the member signals.

This section covers group buses that are used primarily for digital buses, but they can also be used to create differential signals for both analog and digital signals. Differential signals are covered in Section 3.4: Differential signals 56.

Create a Group Bus from existing signals (Most Common):

- Select the signal names in order from lowest bit to highest bit of the bus.
- Click on the **Add Bus** button on the left-hand side of the button bar this will open the *Choose Bus Type* dialog.
- Choose the **Group Bus**



Choose a bus type to add:
[✓ Group Bus] Simulated Bus

Create a Group Bus and it's member signals:

- Make sure that no signals are selected (clear selected signals by clicking in the *Diagram* window), then click the **Add Bus** button to open the *Add Bus* dialog.
- Type the name of the bus into the **Name** box. The member signals will be named the same name as the bus plus their bit index (e.g. BUS0_0, BUS0_1, etc.). If you would prefer the member signals be named as BUS0[0], BUS0[1], etc, first create a virtual bus and convert it to a group bus.
- Select the **Group Bus** radio button, then type in the **MSB** and **LSB** to set the size of the bus

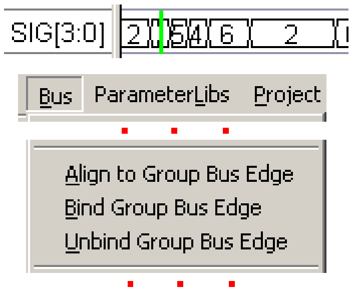
Editing Techniques for Group Bus:

If you are doing a lot of drawing, consider converting to a Virtual Bus which is a lot faster to work with than group buses.

- Draw waveforms on either the Bus signal or on the member signals.
- Double click on a Bus segment to open the *Edit Bus State* dialog and type in either a **Binary** or **Hex** field value. (Virtual state is not used for group buses, because the member signals have to have defined single bit values).
  - 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.
  - **Clear Red Events:** Editing group buses with the Hex button often cause lots of small red rectangles or spikes to appear on the member signals. These mark and preserve the edges on the member signals that now have same state on both sides of the edge. To remove the red marks and merge all adjacent same-state segments, select the **Edit > Clear Red Events** menu.

When you manually draw on the member signals of a group bus, slight differences in the edge times of the member signals will cause multiple edges to be created on the group bus. The Align and Bind features can be used to clean up the edges of the group bus.

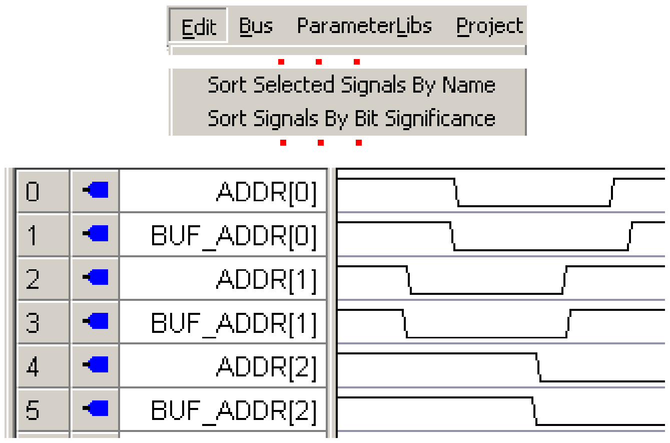
- Click on the bus edge to select it, so that the **Bus > Align**, **Bind**, and **Unbind** menus are activated, then choose one of the menus to perform the action.



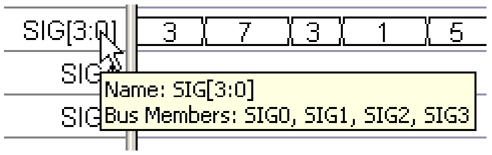
- **Align to Group Bus Edge** will move all the of the nearby member signal transitions (within 10 pixels) to the exact time of the selected bus edge. Make sure to align the edges before adding a delay to a group bus edge, so that all the member signals will also be bound by the delay.
- **Bind Group Bus Edge** will group together all the member signal transitions at the selected bus edge time, so that whenever a transition is moved (on any member signal), all the bound transitions will also move.
- **Unbind Group Bus Edge** menu will ungroup the binded edges so that if a member signal's transition moves, the rest of the transitions at the same time do not move.
- **Delete** a Group Bus by selecting its name and pressing the **Delete** key or by choosing the **Bus > Expand & Delete Bus** menu. Deleting a bus will also delete any parameters attached directly to the bus. The menu method also shows any member signals which might have been hidden.

Display Techniques for Group Buses:

- **Display Hex or Binary data** by double clicking on the group bus name to open the *Signal Properties* dialog and choosing either **Hex** or **Binary** from the **Radix** box.
- **Hide Member Signals** by selecting their names and choosing **View > Hide Signals**.
- To hide member signals automatically during the creation of the bus, choose the **Options > Drawing Preferences** menu option to open a dialog, and check **Hide signals on bus merge** control.
- If you need to visually compare the member signals on two different group buses, you can use the **Edit > Sort Signals by Bit Significance** menu to reorder the signals.
- Note in this example that the individual bits of two group busses are interleaved.



- Display the member signals of a Group Bus in a tooltip by using the mouse to hover over the bus name.



3.3 Simulated Buses

Simulated buses are similar to Group buses in that they have member signals. However, unlike a group bus, the simulated bus is exported to VHDL and Verilog along with its member signals. Simulated Buses were created for the test bench products so that both the bus and the individual signals could be passed into models as needed. A simulated bus is automatically defined as a concatenation of the member signals.

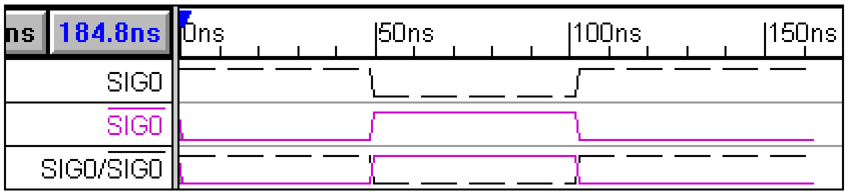
Create a Simulated Bus:

- Click the **Add Bus** button to open the *Add Bus* dialog.
- Select the **Simulated Bus** radio button, and type in a **Name**, an **MSB**, and an **LSB** into the corresponding boxes and close the dialog to place the new bus and its member signals in the timing diagram.

For more information on simulated signals see Chapter 4: Simulated Signals - Boolean Equations 59. For a quick look at the definition of the simulated bus, double click on the bus name to open the *Signals Properties* dialog and view the **Boolean Equation** that defines the bus. For example a simulated bus looks like {BUS2_3,BUS2_2,BUS2_1,BUS2_0}.

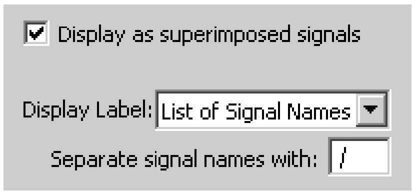
3.4 Differential Signals

Differential signals are two-bit group busses with the display set to superimposed the member signals on top of each other. Each of the original member signals' formatting is displayed by the differential signal. The original signals can be hidden so that the final diagram will only display the resulting differential signal.



Create a differential signal:

- Draw two individual signals. Make one signal a different color or line type by right clicking on the signal name and choosing **Change Signal Color** or **Change Signal line type** from the context menu.
- Create a group bus by first selecting the two signal names and pressing the **Add Bus** button. When the dialog opens select the **Group Bus** radio button.
- Double click on the new group bus signal to open the *Signal Properties* dialog. Since the new signal is a bus the center of the dialog will have the following options displayed:



☑ Display as superimposed signals

Display Label: List of Signal Names

Separate signal names with: /

- Check the **Display as superimposed signals** to have the signals displayed on top of each other instead of the normal hex display for group buses.
- From the **Display Label** drop down, you can choose **List of Signal Names** to change the normal group bus name to a list of signal names. The separation character is controlled by the

text in the **Separate signal names with** box.

3.5 Converting Between Bus and Signal Types

Virtual bus and Group buses can be converted into one another. Also signals can converted in to a bus with the same name.

Convert between a Virtual bus and a Group Bus:

- Select a bus name, then right-click and choose **Group Bus <-> Virtual Bus** from the context menu, OR choose the **Bus > Group Bus <-> Virtual Bus** from the main menu.

Convert Signals with the same name to a Bus:

Single bit signals that have the same signal name except for brackets defining the bit position can be combined into buses (SIG1[0], SIG1[1], SIG1[2] can be combined).

- To operate on the whole diagram, make sure no signal names are selected. To operate on a portion of the diagram, select all the signals that you want to search through.
- Select the **Bus > Convert signals to buses** menu option. to convert all the same name signals.

Convert Signals with Arbitrary names to a Bus:

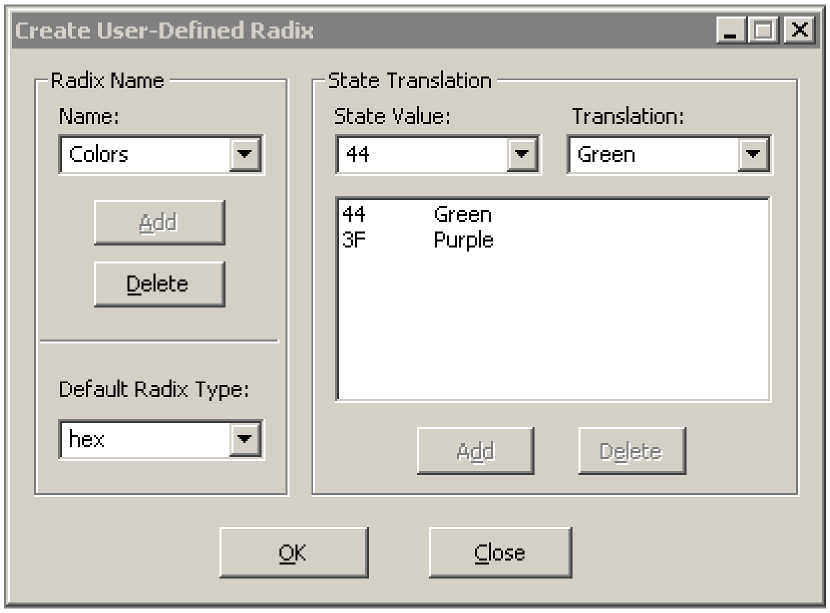
If the names of the signals to be converted do not fit the bracket pattern matching, you can always combine them manually into a group bus using the technique **Create a Group Bus from existing signals** as discussed in Section 3.2: Group Buses 53. This group bus can then be converted to a virtual bus using the **Group Bus <-> Virtual Bus** function.

3.6 Symbolic State Names

Buses can display and export symbolic names instead of the normal numeric state values.

Display Symbolic Names with a User Defined Radix:

- Select the **Bus > Create User-Defined Radix** menu to open the *Create User-Defined Radix* dialog.



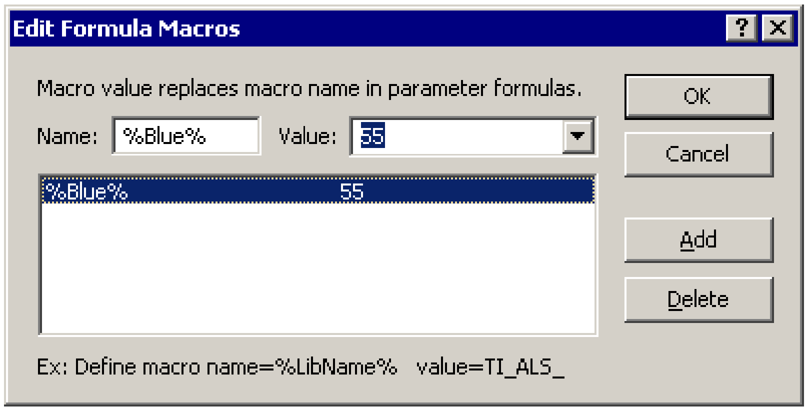
44	Green
3F	Purple

- Add a **Radix Name** that will define the class of values (like Colors).
- Add **State Value** and **Translation** pairs (like 3F Purple), and then close the dialog.
- Double click on a signal name to open the *Signal Properties* dialog and set the Radix of the signal to the Radix that you just created. Now all the virtual values that match a *state value* will display the *translation* instead of the numerical state value.

Exporting Symbolic Bus Values:

Symbolic names for bus state values can be exported to Verilog and System Verilog.

- Select the **ParameterLibs > Macro Substitution List** menu to open the *Edit Formula Macros* dialog and define Name and Value pairs. For Example: **Blue 55**.



Macro value replaces macro name in parameter formulas.

Name: %Blue% Value: 55

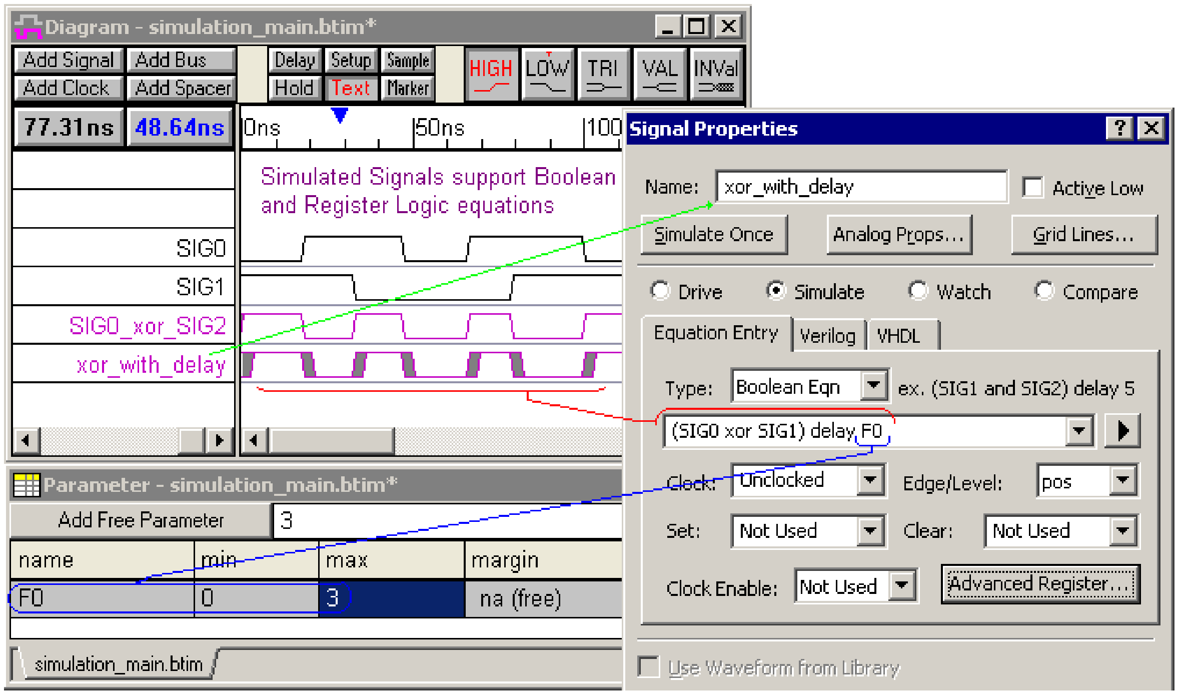
%Blue%	55

Ex: Define macro name=%LibName% value=TI_ALS_

- Next, double click on a waveform segment to open the *Edit Bus State* dialog and enter the **back tick** (not the single quote) and macro name into the **Virtual** box. For Example: **`Blue**.
- When exporting to Verilog, there will be a **`define Blue (55)** at the top of the file, and down in the stimulus generation section, **`Blue** will be used instead of 55 state value.

Chapter 4: Simulated Signals and VHDL/Verilog Export

SynaptiCAD's simulation-enabled products (like WaveFormer Pro and DataSheetPro) contain a built-in Interactive HDL Simulator that is capable of simulating Boolean and registered logic equations. The simulator is interactive, because changes to the input waveforms cause the simulated waveforms to redraw. This feature greatly reduces the amount of time needed to draw a timing diagram, especially one that models gate level circuits.



Simulated Signals support Boolean and Register Logic equations

Add Free Parameter: 3

name	min	max	margin
F0	0	3	na (free)

Name: xor_with_delay

Drive [✓ Simulate] Watch Compare

Type: Boolean Eqn

(SIG0 xor SIG1) delay F0

Clock: Unclocked

Edge/Level: pos

Set: Not Used

Clear: Not Used

Clock Enable: Not Used

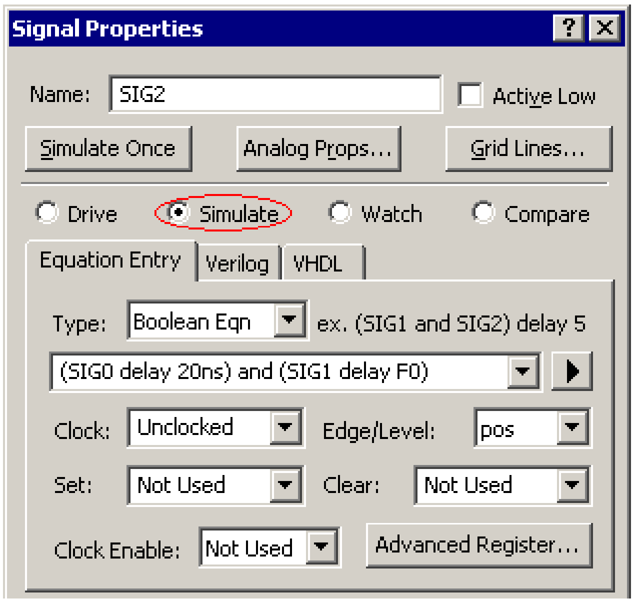
Products like WaveFormer Pro that have export enabled, and also generate VHDL and Verilog stimulus models from the timing diagram. All of the signal types are language independent so the same timing diagram can be used to generate models for both languages.

4.1 Boolean Equations with Delays

Signals can be defined using a Boolean equation of other signals in the diagram. The embedded simulation engine properly simulates multi-bit signals. The simulator also uses true min/max timing models so that the resulting simulated signals can propagate uncertainty delay through the timing diagram.

Creating a Simulated Signal:

- Double click on a signal name to open the *Signal Properties* dialog.
- Select the **Simulate** radio button to allow the signal to be re-simulated each time an input signal is changed.
- Enter a Boolean equation and check any other attributes for the simulation. Section 4.2 [62] covers the Clock and Register features.
- The signal will start to simulate as soon as an equation is entered and it will draw purple waveforms on the simulated signals. If the waveform is grey there is an error in the equation, see the error section below.


Signal Properties

Name: SIG2 ☐ Active Low

Simulate Once | Analog Props... | Grid Lines...

Drive [✓ Simulate] Watch Compare

Equation Entry | Verilog | VHDL

Type: Boolean Eqn ex. (SIG1 and SIG2) delay 5

(SIG0 delay 20ns) and (SIG1 delay F0)

Clock: Unclocked Edge/Level: pos

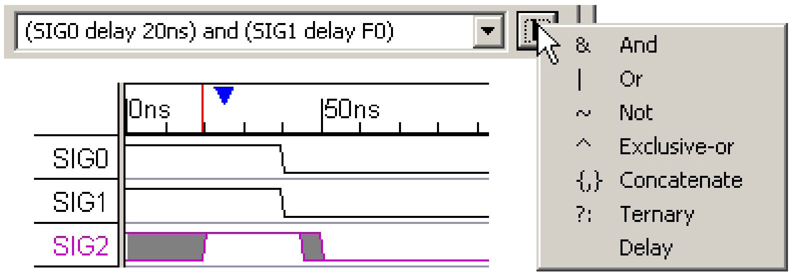
Set: Not Used Clear: Not Used

Clock Enable: Not Used Advanced Register...

Syntax

The Boolean Equation edit box accepts Boolean equations in VHDL, Verilog, and SynaptiCAD's enhanced equation syntax. All signal names are case sensitive.

- The quick fill box contains all of the SynaptiCAD operators.
- The **delay** operator takes a signal on the left and a time or parameter name on the right. If a parameter is entered then the equation will simulate using true min/max timing and puts grey uncertainty regions on the simulated signal.


(SIG0 delay 20ns) and (SIG1 delay F0)

&	And
|	Or
~	Not
^	Exclusive-or
{,}	Concatenate
?:	Ternary
	Delay

- The **delay** operator can also be made to simulate min-only or max-only simulations by selecting the **Options > Diagram Simulation Preferences** menu to open the dialog and then choosing the appropriate setting from the **Timing Model** list.

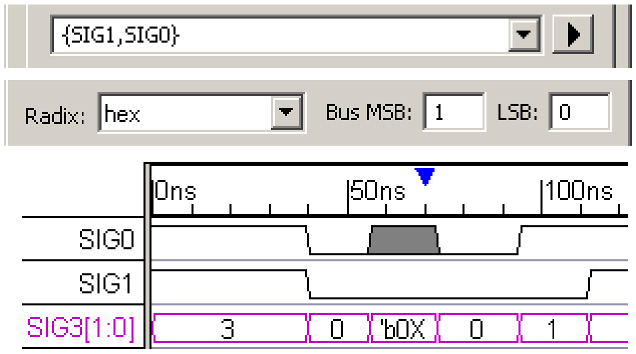
Below are some examples of Boolean equations:

- A 3-input AND gate with a no delay `SIG0 and SIG1 and SIG3`
- A 3-input AND gate with a 20ns delay `(SIG0 and SIG1 and SIG3) delay 20ns`

• A 2-input AND gate using a delay parameter to define the delay time. Each edge of the simulated signal will have a gray uncertainty region that is the difference between the min and max times of GateDelay.	`(SIG0 and SIG1) delay GateDelay`
• A Tristate Gate	`EnableSig ? SIG0 : 'bz`
• A 2-1 MUX	`S0 ? SIG0 : SIG1`
• A 4-1 MUX	`S1?(S0?SIG0:SIG1):(S0?SIG3:SIG2)`
• Signal Concatenation (Note: the simulated signal must have a proper MSB size to handle the result)	`{SIG0, SIG1}`
• Concatenate bit slices	`{SIG0[3:0], SIG1[7:4]}`

Multi-bit Simulation is Automatically supported

- Set the **MSB** and **LSB** in the *Signal Properties* dialog to model multi-bit simulated signals.
- If a signal has a non-binary radix, and one of the bits on that signal is in an unknown state (X), then that segment will be displayed as a binary value.



{SIG1,SIG0}

Radix: hex Bus MSB: 1 LSB: 0

- For example in the picture, SIG3 has a radix of hex and most of the value are displayed in hex. However, one of the segments has an unknown state, and that state is displayed in binary since 'b0X is not a valid hexadecimal value.

Errors

There is an error status bar on the bottom right of the big window.

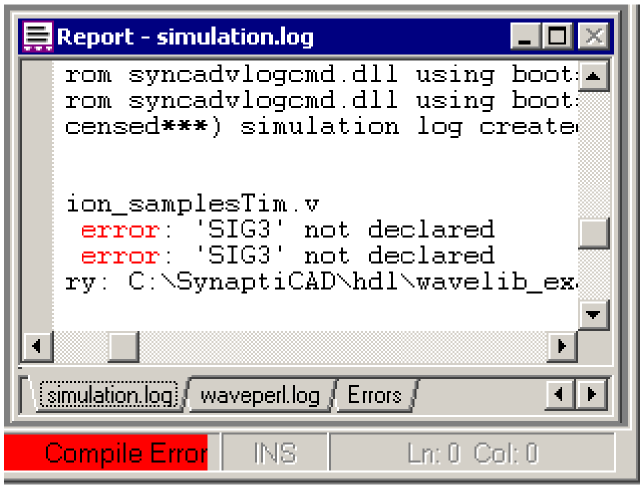
- **Simulation Inactive** means that there are no signals with the type **Simulate**

Simulation Inactive | INS | Ln: 0 Col: 0

Simulation Good | INS | Ln: 0 Col: 0

- **Simulation Good** shows that all equations are simulating without errors

- **Compile Error** shows that something is wrong with an equation.
- The **Simulation.log** file displayed in the Report window shows simulation results and any simulation errors.
- The **Error** tab shows simulation errors in a condensed tabular form.

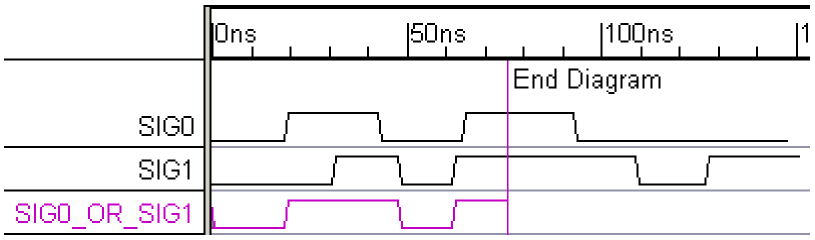


```
rom syncadvlogcmd.dll using boot:
rom syncadvlogcmd.dll using boot:
censed***) simulation log create

ion_samplesTim.v
 error: 'SIG3' not declared
 error: 'SIG3' not declared
ry: C:\SynaptiCAD\hdl\wavelib_ex.
```

Stopping a Simulation with an End Diagram Marker

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.

- To force the simulation to stop at a specific time, add a **Marker** to the diagram and set its type to be **End Diagram**. See Section 6.4: Marker Lines 96.



Use care when attaching graphical parameters to a simulated signal:

When a Boolean equation is re-simulated, the signal's waveform is cleared of event edges. The program attempts to preserve the attachment of parameters and text objects on signal edges, but the addition or loss of edges during the re-simulation may move these objects.

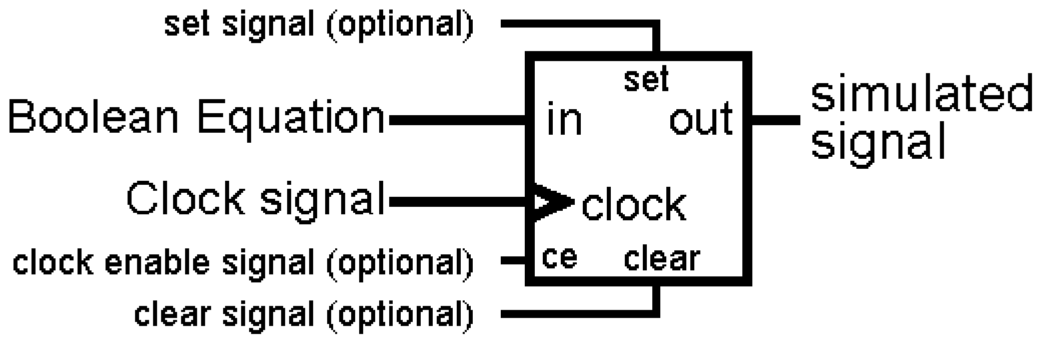
Partially Simulated Signals:

Sometimes it is useful to use an equation to draw the initial waveform, then return it to a "drawn" waveform that allows manual editing.

- To simulate once, enter an equation, leave the button on **Drive**, and press the **Simulate Once** button. This will simulate the waveform using the current inputs but the waveform will be black to indicate it can be edited.

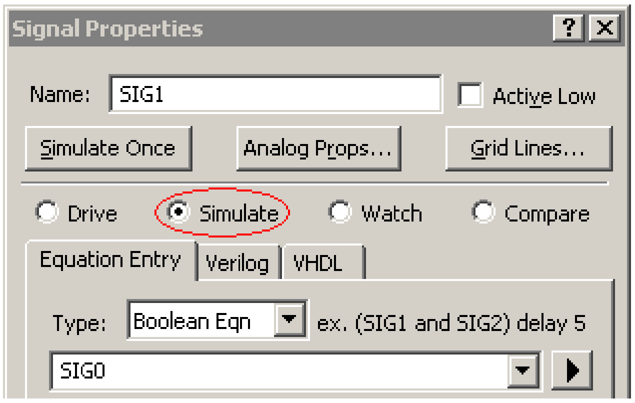
4.2 Register and Latch Equations

A signal can be defined as a register or latched equation similar to a cell in an FPGA. The Boolean equation is the input to a register or latch and the model is determined by the options that you choose. These signals will properly export as VHDL and Verilog models rather than just straight waveform stimulus. The following is a diagram of the model defined in the *Signals Properties* dialog.



Using a Registered or latched Equation:

- Double-click on a signal name to open the *Signal Properties* dialog and enter an equation into the **Boolean equation** box and select the **Simulate** button.
- The Boolean equation can either be the name of the input signal or an entire equation.



Signal Properties

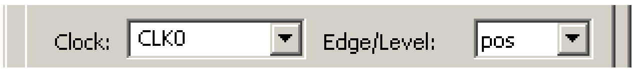
Name: SIG1 — Active Low: unchecked

Drive ( ) [✓ Simulate] Watch ( ) Compare ( )

Type: Boolean Eqn — ex. (SIG1 and SIG2) delay 5

SIG0

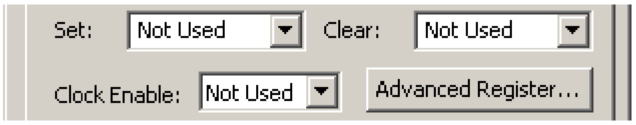
- To activate the register or latch model, you must choose a **Clock** signal and an **Edge/Level** type. The clock can be any signal in the diagram.



Clock: CLK0 — Edge/Level: pos

- For a **Register** circuit choose **neg**, **pos**, or **both** to define the triggering edge.
- For a **Latch** circuit choose either **low** or **high** level latching

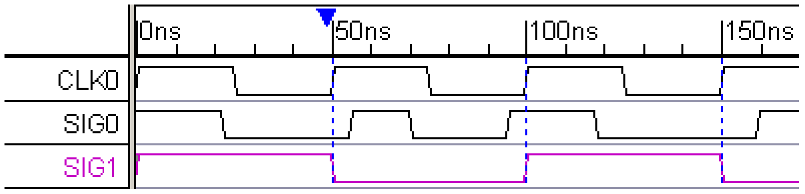
- The **Set**, **Clear**, and **Clock Enable** are optional signals. If "Not Used" is chosen for a line, then that line is not modeled.



Set: Not Used — Clear: Not Used

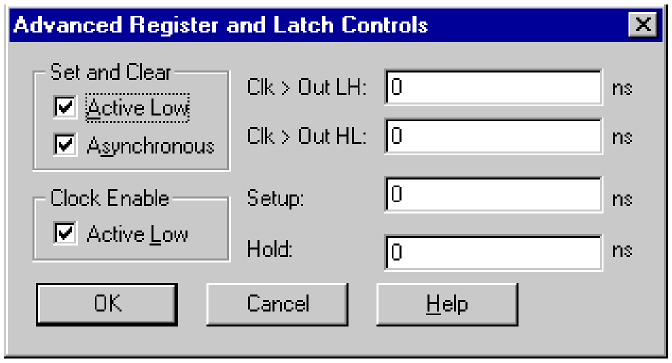
Clock Enable: Not Used

- Below is an example. SIG1 is the output of a positive edge triggered register. The CLK0 is the input clock and SIG0 is the signal that is being registered. The blue lines are grid lines placed on the positive edge to the clock to show the sampling points (See Section 2.3: Grid Lines on Clocks and Signals 44).



Copyright © 2011, SynaptiCAD Sales, Inc.

Altering the Register or Latch Models:



Set and Clear: Active Low [✓], Asynchronous [✓]
Clock Enable: Active Low [✓]
Clk > Out LH: 0 ns
Clk > Out HL: 0 ns
Setup: 0 ns
Hold: 0 ns

- Pressing the **Advanced Register** button opens the *Advanced Register and Latch Controls* dialog and has the controls for the timing options for this signal's register or latch model.
- The **Set and Clear** lines can be active low or active high. The can also be asynchronous so that a change on the line immediately effects the output. Or they can be synchronous so that the output does not change until the next active level of the clock.
- **Clock Enable** line can be either active high or low.
- **Clock to Out** boxes specify the delay from the triggering of the clock signal to a change on the output edge. This setting supports both a Low to High model and a High to Low model.
- **Setup** specifies the time for which the input must be stable before the clock-triggering event. If a min/max timing parameter is entered, Setup will use the min time. Any violations of this setup time will be reported to the simulation log in the Report window.
- **Hold** specifies the time for which the input must remain stable after the clock-triggering event. If a min/max timing parameter is entered, Hold will use the min time. Any violations of this hold time will be reported to the simulation log in the Report window.
- The global defaults for this dialog can be defined using the **Options > Diagram Simulation Preferences** menu.

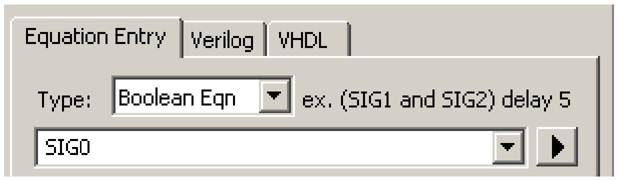
Replacing SynaptiCAD's Register or Latch Models:

The files **SynaptiCAD\hdl\wavelib_*ModelType*.v** contain the generic register and latch models.The *ModelType* is determined by the **Flip-flop library** box **Options>Diagram Simulation Preferences** dialog. Be very careful and make copies if choose to edit or replace these models.

4.3 Direct Signal Code

The **Verilog** and **VHDL** tabs in the *Signal Properties* dialog allow you to view and edit the generated code for a specific signal.

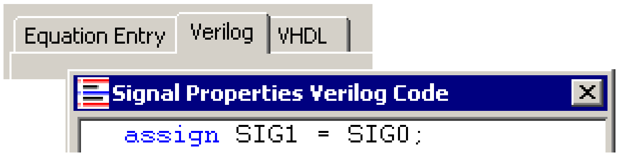
To View or Edit the Simulated Signals Code:



Type: Boolean Eqn ex. (SIG1 and SIG2) delay 5
SIG0

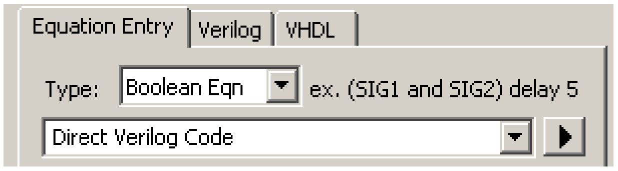
- Double click on a simulated signal to open the *Signal Properties* dialog. The **Equation Entry** tab shows the settings for the simulated signal

- Press the **Verilog** or **VHDL** tab to open an editor window with the generated code displayed



```
assign SIG1 = SIG0;
```

- If you make changes to the generated code, they will be saved when you close the window or click to another tab. After the changes, the Equation Entry will then be disabled and a **Direct Code** message will be displayed in the equation box.



- WaveFormer Pro by default is configured to simulate Verilog code using its internal simulator. If you make direct code changes to the VHDL code, then you will also have to provide and setup a VHDL simulator as discussed in Section 4.5 HDL Code Generation Settings 69.

Tips on writing direct code:

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. **Signals are always treated as wires during Verilog 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 would be written as:

```
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
```

   Notice that you do not have to declare the WaveFormer signal **SIG0** because WaveFormer automatically creates it. However, you do have to declare the temporary reg **tempSIG0** because WaveFormer doesn't know anything about it.

4.4 Export VHDL and Verilog test benches

SynaptiCAD offers several different levels of test bench generation. This section covers the features in WaveFormer Pro and WaveFormer Lite which generates VHDL and Verilog stimulus models from a single timing diagram. The generated files are normally used with external simulators. SynaptiCAD also offers more advanced code generation in the following products:

- **Reactive Test Bench Generation Option** upgrades WaveFormer, VeriLogger, and BugHunter so that they can generate single diagram test benches that react to the model under test during simulation and produce pass or fail reports. See the Reactive Test Bench manual for more information about this feature.

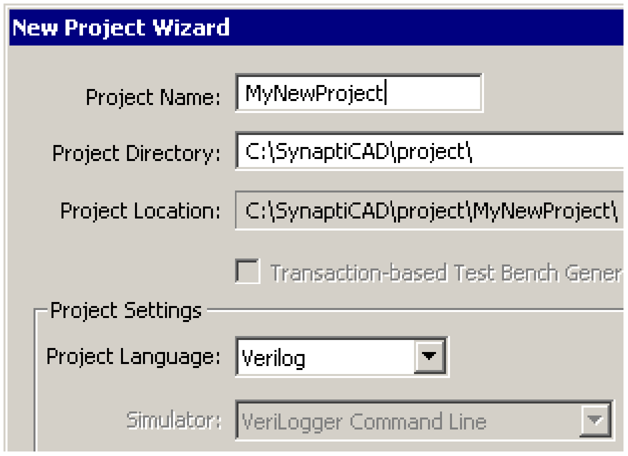
- **TestBencher Pro** generates multi-diagram, self-testing, reactive test benches. Perfect for building bus-functional models of microprocessor and bus interfaces. The TestBencher Manual describes this type of generation.
- **VeriLogger Pro and BugHunter Pro** also generate interactive stimulus test benches (like WaveFormer) except that the testbench is tightly integrated into the simulation debugger and allows for very quick testing of models. Chapter 3: Waveforms and Test Bench Generation of the ***BugHunter & VeriLogger Manual*** describes this type of generation.

Import or setup the Signal names and types

In order for the exported testbench model to hook up to your model under test, the signal names (including type, size and direction) must match. You can either have WaveFormer parse and extract this information (auto-extraction requires Reactive Test Bench option) or you can set it manually using the *Signal Properties* dialog.

Auto-extraction Steps (Reactive Test Bench option required)

- To extract the information from the models under test, first create a project.
- Select the **Project > New Project** menu to open the *New Project Wizard* dialog, enter a **project name**, and set the **project language**.
- Pressing the **finish** button will cause all the currently open windows to close and a project window to open.


New Project Wizard

Project Name: MyNewProject

Project Directory: C:\SynaptiCAD\project\

Project Location: C:\SynaptiCAD\project\MyNewProject\

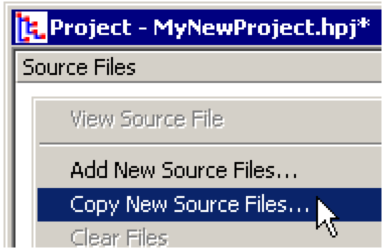
Transaction-based Test Bench Gener

Project Settings

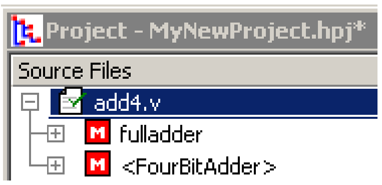
Project Language: Verilog

Simulator: VeriLogger Command Line

- Next, add the model files to the project, by right clicking in the Project Window and choosing one of the **New Source File** menus. **Copy** first copies the model file to the project directory and then adds it to the project. **Add** just adds the file to the project without moving it.

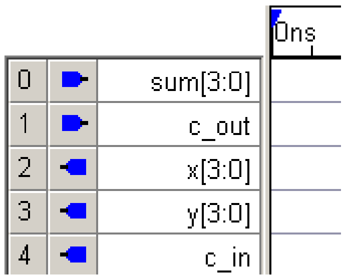


- Press the **Extract MUT Ports into diagram** button on the main button bar. This will cause WaveFormer to parse the files in the project and populate the project tree so that you can view a hierarchical view of the model under test.



Project - MyNewProject.hpj*
Source Files
add4.v
fulladder
<FourBitAdder>

- The Extract MUT Ports button will also populate a timing diagram with the ports of the top level module. Output signals drive the model under test and have icons that point to the left. Input signals represent signals with data coming back from the model under test and have icons that point to the right.



0ns

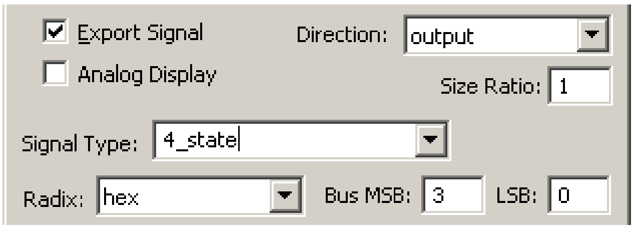
0	sum[3:0]
1	c_out
2	x[3:0]
3	y[3:0]
4	c_in

- You are now ready to draw your testbench waveforms for the signals in the diagram window.

Or manually enter signal type information:

If you add the signals manually, the following settings will need to be set for each signal:

- Double click on a signal's name to open the *Signal Properties* dialog and edit the **name** and **MSB/LSB**.
- Check the **Export Signal** box so that this signal will be included in stimulus generation.



[✓] Export Signal
Direction: output
[ ] Analog Display
Size Ratio: 1
Signal Type: 4_state
Radix: hex
Bus MSB: 3
LSB: 0

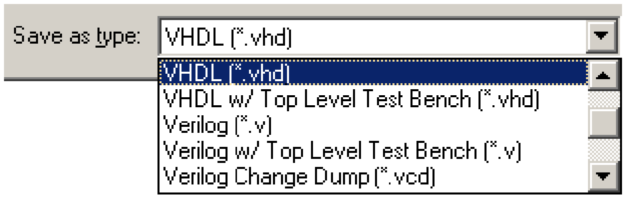
- Use the **Direction** box to choose **output** so that the signal will drive the model under test. The other directions are for use with more advanced test bench generation of Reactive Test Bench or TestBencher Pro.
- Choose a **Signal Type** or type in a user-defined type into that box. The standard types are language independent, so the same timing diagram can be used to generate both VHDL and Verilog testbench models. The following are the mappings between the types:

SynaptiCAD	Verilog	VHDL
4 state	reg	std logic
4 state vector	reg	std logic vector
bool	reg	boolean
2 state	reg	bit
2 state vector	reg	bit vector
byte	reg	bit vector
int	integer	integer
unsigned int	integer	natural
real	real	real
fixed len string	reg	string
variable len string		
time	time	time
event	event	
std logic		std logic
std logic vector		std logic vector
std ulogic		std ulogic
std ulogic vector		std ulogic vector
signed logic		signed

unsigned logic		unsigned
actel_current_delta	reg	std_logic
actel_temperature	reg	std_logic
actel_voltage	reg	std_logic
actel_voltage_common	reg	std_logic
actel_voltage_delta	reg	std_logic

To export a timing diagram:

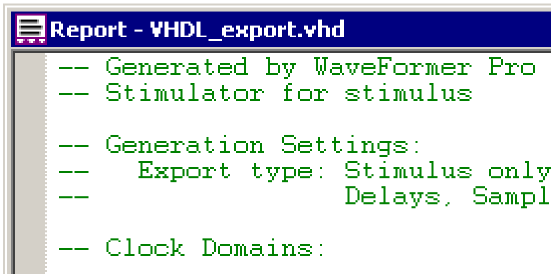
- Select the **Import/ Export > Export Timing Diagrams As** menu option to open a special version of the *Save As* dialog which remembers the file type of the last file exported.
- Select a VHDL or Verilog export format from the **Save as type** box, enter a file name, and click Save button to generate the file.
- The generated file will open as a tab in the Report window so you can quickly see how changes in the diagram affect the generated code.


Save as type:
VHDL (*.vhd)
VHDL (*.vhd)
VHDL w/ Top Level Test Bench (*.vhd)
Verilog (*.v)
Verilog w/ Top Level Test Bench (*.v)
Verilog Change Dump (*.vcd)

The plain VHDL or Verilog formats generates a stimulus module that can be included into a user-written top level test bench.

The **Top Level Test Bench** types instantiates the model in the *Project* window and the stimulus model within a top-level model. This top-level module can then be simulated without any additional setup.

The VCD format generates a stimulus file that can be read by other EDA tools and test equipment (not a test bench model).

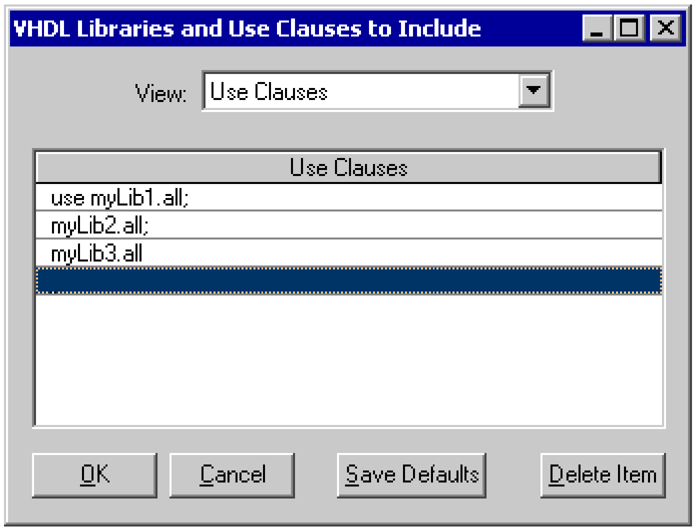


```
-- Generated by WaveFormer Pro
-- Stimulator for stimulus

-- Generation Settings:
--    Export type: Stimulus only
--                 Delays, Sampl

-- Clock Domains:
```

VHDL Library and Use Clause statement support:

If Libraries or Use Clauses need to be added to the generated model, then specify them in the *VHDL Libraries and Use Clauses to Include* dialog.

- Choose the **Options > VHDL Libraries and Use Clauses** menu to open the dialog. Changes made in this dialog will be applied to all diagrams that are exported to VHDL.
- From the **View** control select either **Use Clauses** or **VHDL Libraries** and enter the information on to a blank row.

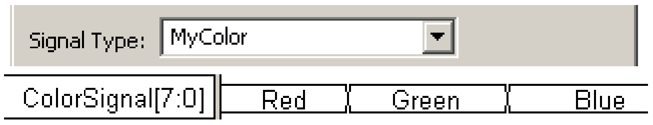


use myLib1.all;
myLib2.all;
myLib3.all

- If the 'USE' is omitted from a use clause, or the 'LIBRARY' from a library statement, WaveFormer will automatically add 'USE' or 'LIBRARY' before the clause in the source file for the diagram. If necessary, WaveFormer will also automatically add semicolons to the end of library includes and use clauses

VHDL User-Defined Types are supported:

- In the *Signal Properties* dialog, type in the user-defined type into the **Signal Type** box.



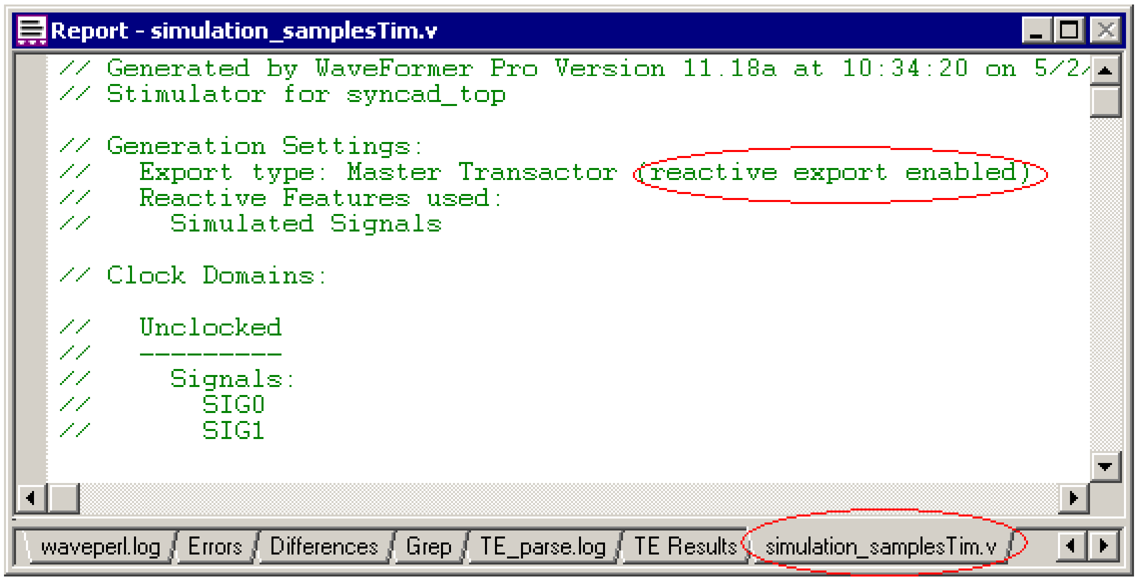
- On the waveform, double click to open the *Edit Bus State* dialog and enter the state values into the **Virtual** box.
- To make the generated code compilable when you include signals with user-defined types, you will need to add a ***use clause*** for the package that defines the type. See the section above this for more information on use clauses.

4.5 HDL Code Generation Settings

WaveFormer Pro generates a Verilog or VHDL model file from the information in each of the signals. Signals of type **Drive** generate stimulus vectors, clocks generate clock models, and simulated signals generate Boolean equations, register, or latch circuits. You can view the model file to get an idea of the code that is generated. The Diagram Simulation Preferences dialog has several controls that affect the type of code generated.

Viewing the Generated file:

- Press on the tab named *timing_diagram_name*.v in the *Report* window to see the code (or .vhd depending on the selected simulator). If you cannot see the tab, then press the right hand arrow next to the tab bar to see more tabs. If you cannot see the Report window, select the **Window > Report** menu option to bring it to the top.



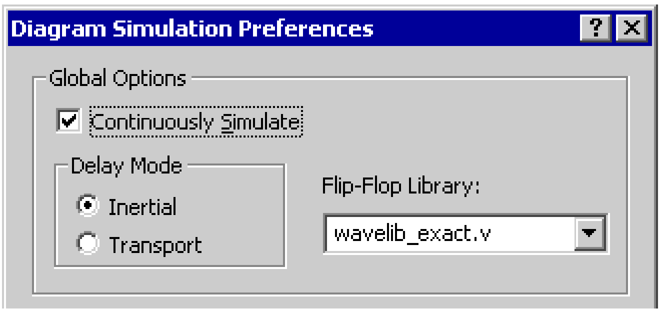
```
// Generated by WaveFormer Pro Version 11.18a at 10:34:20 on 5/2/
// Stimulator for syncad_top

// Generation Settings:
//   Export type: Master Transactor (reactive export enabled)
//   Reactive Features used:
//     Simulated Signals

// Clock Domains:

//   Unclocked
//   ---------
//     Signals:
//       SIG0
//       SIG1
```

- The top of the file shows which options are enabled for the program. In the above example, the comment indicates that the *Reactive Export* feature is enabled. The next comments show which signals are in each clocking domain (set by the Clock dropdown in the *Signal Properties* dialog). A signal's clock domain determines when state changes are output on driven signals (by default, signals are "unclocked" and driven after time delays.

Changing the type of code generated:

The *Diagram Simulation Preferences* dialog controls what code is generated for timing diagram simulations.

- Select the **Options > Diagram Simulation Preferences** menu to open the dialog.



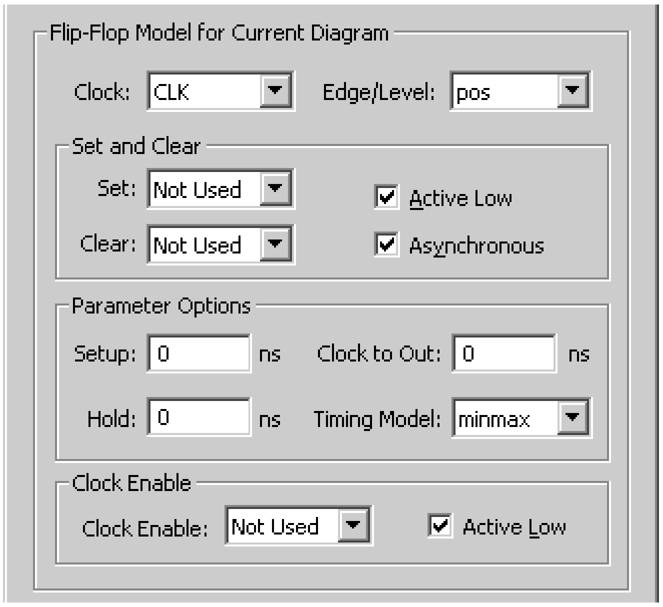
Global Options
[✓] Continuously Simulate
Delay Mode
[● Inertial] ○ Transport
Flip-Flop Library: wavelib_exact.v

- The Global Options affects simulation of all timing diagrams in the editor.
- The **Inertial** delay mode causes pulses that are smaller than delay values to disappear.
- The **Transport** delay mode causes all pulses to be transmitted.

- The **Flip-Flop Library** allows you to specify the parts library that is used for the simulation. This feature is for advanced users who wish to modify the way registers and latches are modeled by the tool.

- The **Continuously Simulate** box, if checked, allows the editor to re-simulate each time an edge is moved. This provides the "Interactive" part of the environment. However, you may wish to turn off this feature if the simulations begin to take a long time to run and you do not want to simulate each time you make a minor change. If this box is unchecked, then you can force a simulation by pushing the **Simulate Diagram** button on the button bar or by pushing the **Simulate Once** button in the *Signal Properties* dialog.

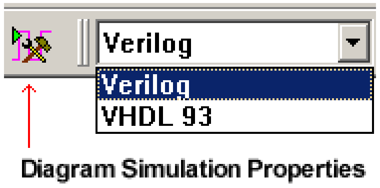
- The **Flip-Flop Model for Current Diagram** area sets the defaults for newly created signals in the current diagram. These controls work the same as the controls in the *Signals Properties* dialog and the *Advanced Register* dialog covered in Section 4.2 62, except that they affect all new signals instead of just the selected signal.


Flip-Flop Model for Current Diagram
Clock: CLK Edge/Level: pos
Set and Clear
Set: Not Used [✓ Active Low]
Clear: Not Used [✓ Asynchronous]
Parameter Options
Setup: 0 ns Clock to Out: 0 ns
Hold: 0 ns Timing Model: minmax
Clock Enable
Clock Enable: Not Used [✓ Active Low]

Changing the generated language and the simulator

WaveFormer Pro's default behavior is to generate a Verilog model file and give it to a special version of SynaptiCAD's VeriLogger Pro to perform the simulation. The version of VeriLogger Pro that ships with WaveFormer is an interpreted simulator that is very fast at starting a simulation and finishing small simulation files. However, the simulator and the type of code generated can be changed.

- To change the generation language, use the drop-down on the button bar. If you choose VHDL, you must also provide a VHDL simulator so that the simulated signals work.



- To change the Simulator, press the Diagram Simulation Properties button to open the dialog.

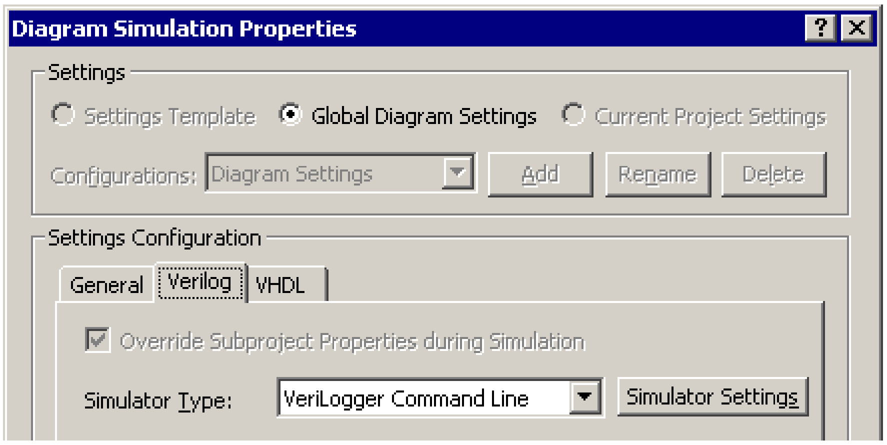
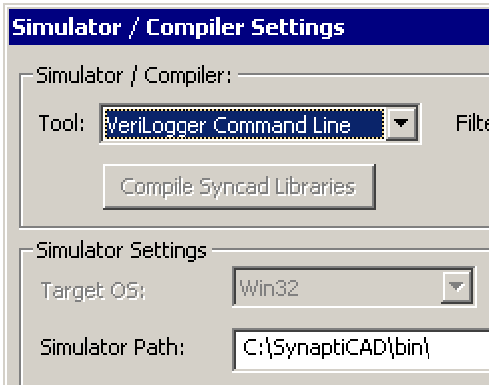


Diagram Simulation Properties
Settings
Settings Template [✓ Global Diagram Settings] Current Project Settings
Configurations: Diagram Settings Add Rename Delete
Settings Configuration
General | Verilog | VHDL
[✓ Override Subproject Properties during Simulation]
Simulator Type: VeriLogger Command Line Simulator Settings

- Press the **Verilog** or **VHDL** tab, and choose a simulator from the **Simulator Type** box.

- If you have never used the new simulator with WaveFormer, press the **Simulator Settings** button to open the *Simulator / Compiler Settings* dialog.
- Use this dialog, to enter the path for the new simulator.


Simulator / Compiler Settings

Simulator / Compiler:

Tool: VeriLogger Command Line

Simulator Settings

Target OS: Win32

Simulator Path: C:\SynaptiCAD\bin\

4.6 Report Window

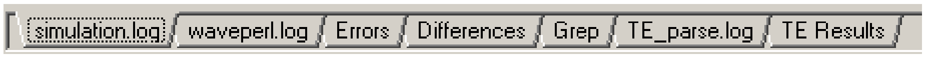
The *Report* window displays error reports and generated code. It is also a general purpose editor and can be used to view and edit text files.

Finding the Report Window

- Select the **Window > Report** menu to bring the Report Window to the front. There will always be a report window open.

Default Tabs

The Report window will open with a variety of tabs. Also each time you export the timing diagram to a different format (like VHDL or SPICE) the generated file will be displayed in a new tab.



- Simulation.log, waveperl.log, and Error tabs show simulation and perl scripting errors.
- Differences shows a tabular form of difference for comparison errors from the Waveform Comparison feature.
- Te_parse.log and TE Results show the errors and results of Temporal equations from the Transaction Tracker feature.

Report Window is also a Text Editor:

To use the *Report window* as an editor use the following Report menu options to open and save new files:

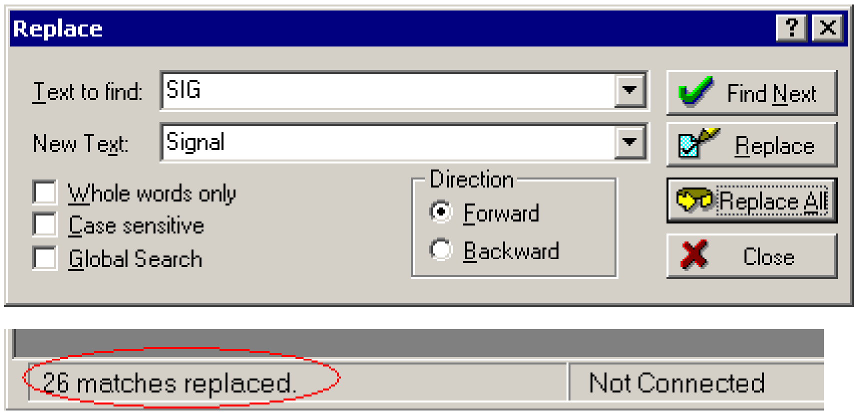
- **Report > Open Report Tab** opens a new file
- **Report > Close Report Tab** closes the displayed tab and file
- **Report > Save Report Tab** saves the displayed file
- **Report > Save Report Tab As** saves the displayed file under a different name.
- **Report > Print Report Tab** prints the displayed 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**.

The *Report* window displays are full-featured editor windows. Some editing features can be reached by right clicking in the report tab and choosing the command.

- Right click in the Report Window and choose **Find** from the context menu to open the *Replace* dialog.
- When a replace is completed, the number of matches is displayed in the bottom left corner of the Main program window.



Below is a complete list of keyboard and mouse commands which are supported by the editor window.

Key	**Purpose**
`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)
Ctrl+G	Jump to line#

Chapter 5: Delay, Setup, & Hold Parameters

Delay, Setup, and Hold parameters actively move and monitor signal transitions. They provide the automatic updates and timing analysis capabilities of the timing diagram editor.

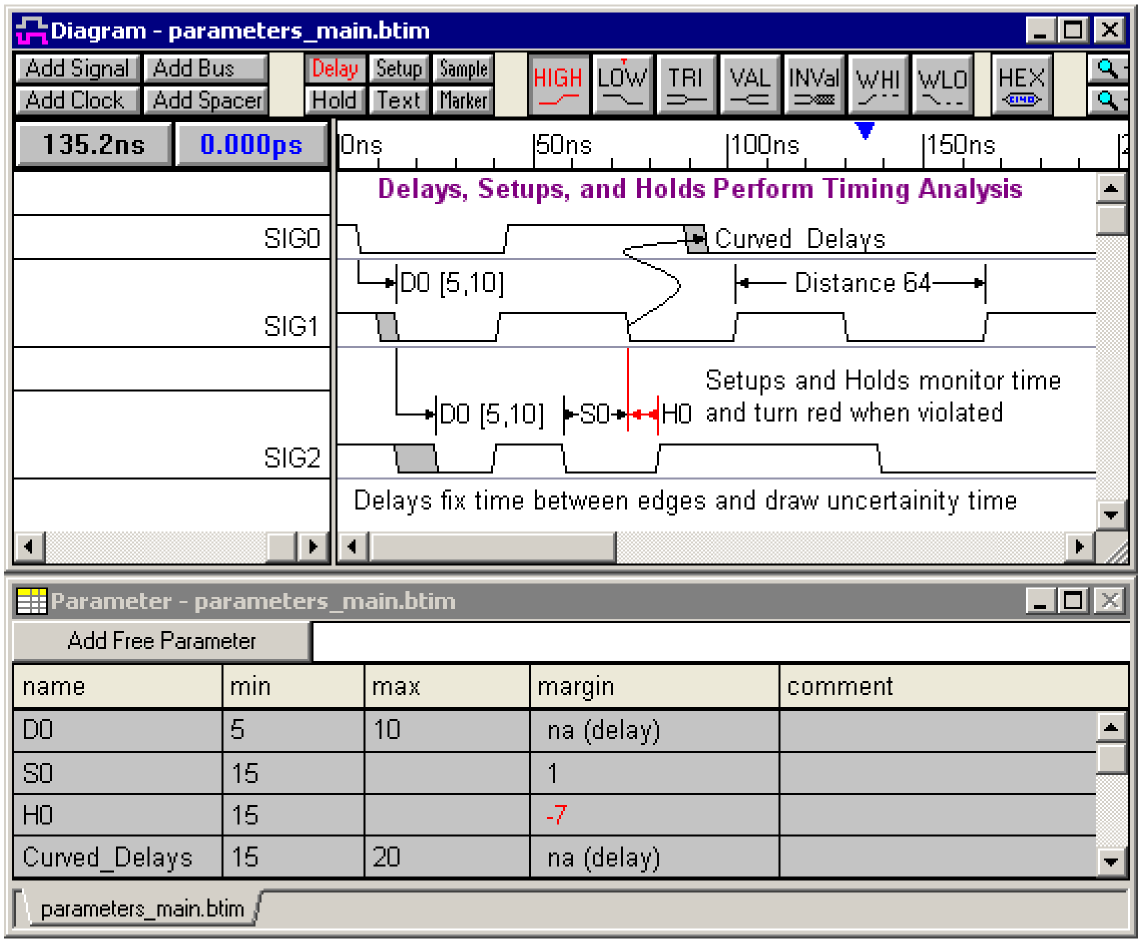


Diagram - parameters_main.btim
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
HIGH
LOW
TRI
VAL
INVal
WHI
WLO
HEX
135.2ns
0.000ps
0ns
50ns
100ns
150ns
Delays, Setups, and Holds Perform Timing Analysis
SIG0
SIG1
SIG2
Curved Delays
D0 [5,10]
Distance 64
Setups and Holds monitor time and turn red when violated
D0 [5,10]
S0
H0
Delays fix time between edges and draw uncertainity time
Parameter - parameters_main.btim
Add Free Parameter
parameters_main.btim

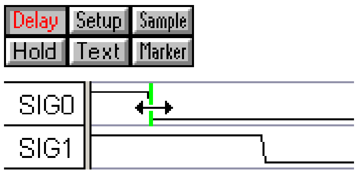
name	min	max	margin	comment
D0	5	10	na (delay)	
S0	15		1	
H0	15		-7	
Curved_Delays	15	20	na (delay)	

5.1 Delays

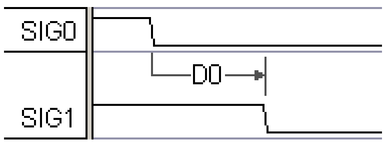
A delay specifies a fixed time or a number of clocking signal edges between two signal transitions. Typically both the **min** and **max** values are used in defining delays. If the min and max values are different, the delayed signal transition will be displayed with a gray region of uncertainty. Delays cannot have circular dependencies - if A delays B, and B delays C, then C cannot delay A.

Adding a Delay:

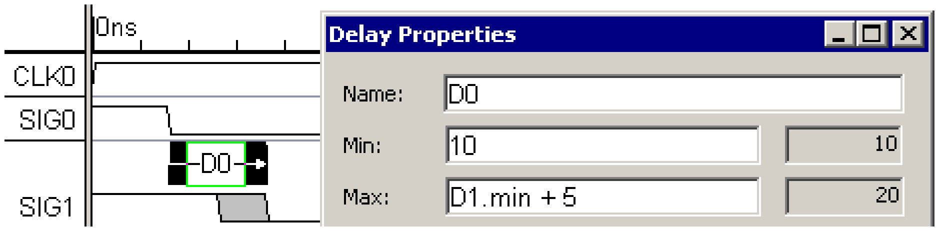
- Press the **Delay** button so that right clicks will add delays.
- Left click on the first transition in time to select it.



- Right click on the second transition to add the delay. See Section 5.4 83 on moving delays after they are drawn.



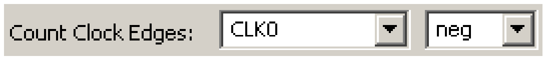
- Double click on the delay to open the *Delay Properties* dialog, and enter the min or max times (or time formulas like **D1.min + 5**). See Section 2.5 Time Formulas for Clocks and Parameters 47 for information on the syntax of the formulas.



Delay Properties

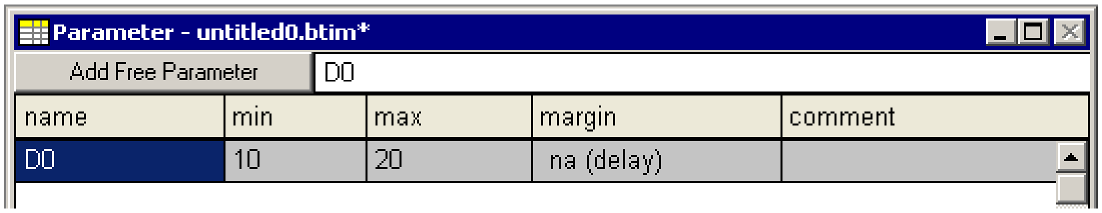
Name:	D0	
Min:	10	10
Max:	D1.min + 5	20

- OPTIONAL: Delays can be clocked instead of time based, by specifying a **clocking signal** and **edge type** (pos, neg, or both) in *Delay Properties* dialog. Once this is done, the **min** and **max** boxes will specify the number of edges instead of fixed times.



Count Clock Edges: CLK0 neg

- Notice that the Delay has also been added to the *Parameter* window. The *margin* field has no meaning for delays and is marked as "na".



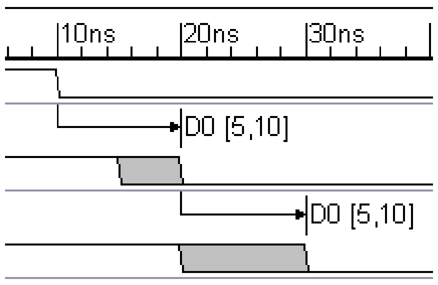
Parameter - untitled0.btim*

Add Free Parameter D0

name	min	max	margin	comment
D0	10	20	na (delay)	

Delay Add Up:

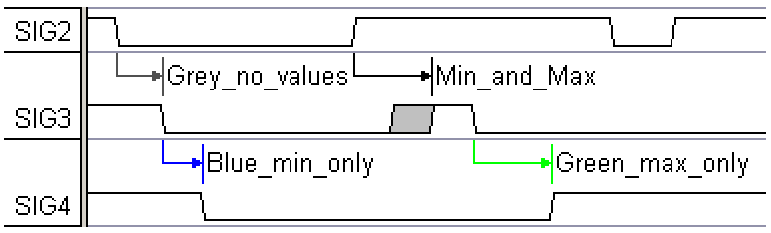
Delays define a timing path through a circuit. As the timing path goes through each gate the delay uncertainty regions will add together. Here both delays have 5ns min and a 10ns max. The first uncertainty region is 5ns wide, and represents the difference between the min and max times of the first delay. The second uncertainty region is 10ns wide. The first edge is set by both delay's min times and the second edge is set by the delay's max times.



Delay Colors and Critical Paths:

Delays are color coded to indicate which edges of a transition they control. Color coding can be turned off for documentation purposes, by un-checking the **View > Show Critical Paths** menu.

- **Black** delays set both the min and max edges of a signal transition.
- **Blue** and **green** indicate min only and max only edge setting.

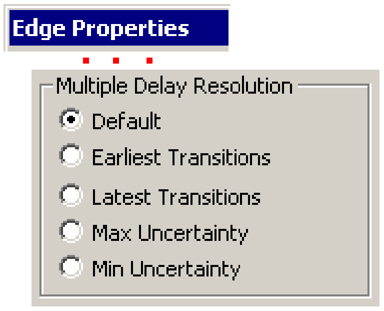


- A **gray** delay sets neither edge of the delayed transition, either because the min/max values are blank, another delay is dominant at the delayed transition, or the delayed edge has been locked[26].

Reconciling multiple delays ending on an edge:

If more than one delay ends on the same transition, then the colors will show which delay is responsible for which transition. To resolve this for a particular edge:

- Double click on the edge to open the *Edge Properties* dialog and choose a **Delay Resolution** setting. The default method is set using the **Options > General Preferences** menu.
- **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.


Edge Properties

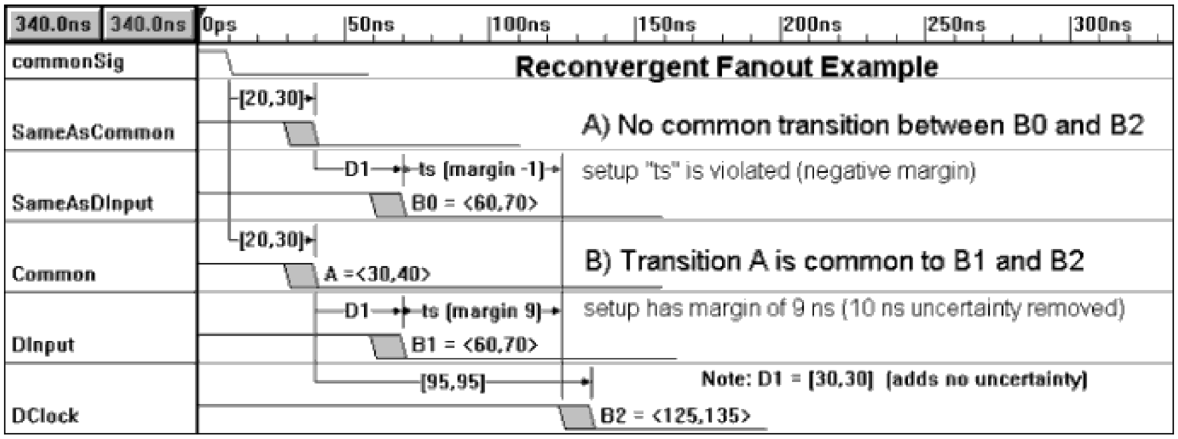
Multiple Delay Resolution
[✓ Default]
Earliest Transitions
Latest Transitions
Max Uncertainty
Min Uncertainty

- **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).

The best way to figure this out is to play with the **Multdely.btim** timing diagram in the **Examples** directory.

Common Delay Removal (Reconvergent fanout):

The timing diagram editor automatically accounts for reconvergent fanout effects, correctly calculating margins and distances for parameters. Reconvergent fanout happens when two *timing paths* share a *common transition.* The uncertainty of the *common transition* should not affect distance and margin constraints between transitions further down on the timing paths, because whenever the *common transition* occurs it occurs at the same time for both timing paths. This effect is called **reconvergent fanout** because it is only important when a signal fans out to multiple gates and the outputs of these gates reconverge as inputs to a common gate such as a data and a clock signal to a register. If the effects of reconvergent fanout were ignored, a good design might seem to violate a timing parameter because the margin calculations were overly pessimistic. 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).

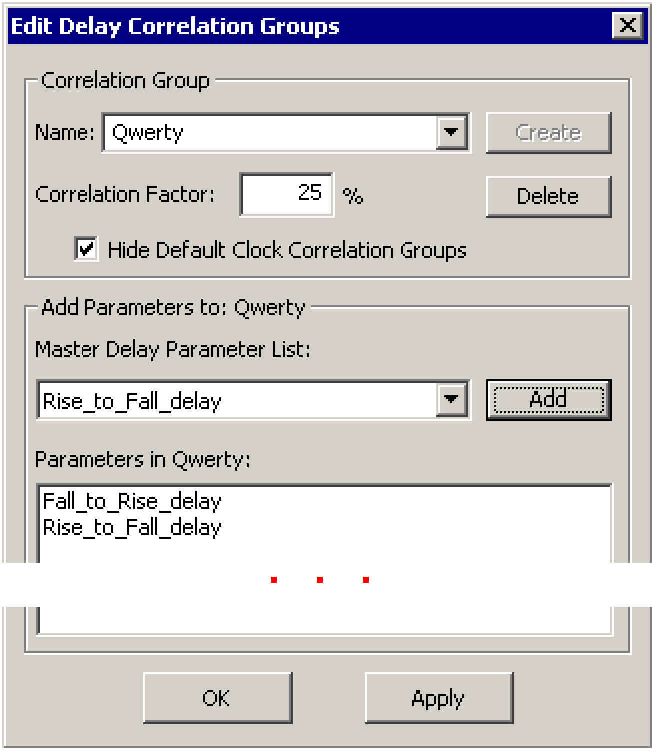

340.0ns
340.0ns
0ps
50ns
100ns
150ns
200ns
250ns
300ns
commonSig
SameAsCommon
SameAsDInput
Common
DInput
DClock
Reconvergent Fanout Example
[20,30]
A) No common transition between B0 and B2
D1
ts (margin -1)
setup "ts" is violated (negative margin)
B0 = <60,70>
[20,30]
B) Transition A is common to B1 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>

Example: A NAND gate has an uncertainty region of 10ns. Two signal paths fanout from the NAND gate, go through some other gates, and then reconverge at the data(DInput) and clock(DClock) 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 affect the difference in arrival times because both signals will start at the same time (somewhere in the uncertainty region of the NAND gate).

Delay Correlation Groups

Even though data sheets for IC's list large min/max uncertainty ranges for each delay, the delays within any particular IC will be much more tightly grouped. To take advantage of this, delays from a particular IC can be added to a correlation group. The editor will then recalculate the ***setup*** and ***hold*** times so that faster circuits can be designed. Once a delay is added to a correlation group the new calculations are automatically done.

- Double-click on a delay to open the *Delay Properties* dialog, then click the **Correlation** button to open the *Edit Delay Correlation Groups* dialog.
- To create a new group, type a group name into the **Name** box and click the **Create** button. To access an existing group, just select it from the drop-down **Name** list.
- In the **Correlation Factor** box, enter the percent correlation for the delays in this group. The percentage must be a whole number between 0 (no correlation) and 100 (fully correlated).
- Next, add delays to the group by selecting them in the **Master Parameter List** and pressing the **Add** button.



Correlation Group

Name: Qwerty

Correlation Factor: 25 %

[✓] Hide Default Clock Correlation Groups

Add Parameters to: Qwerty

Master Delay Parameter List:

Rise_to_Fall_delay

Parameters in Qwerty:

Fall_to_Rise_delay
Rise_to_Fall_delay

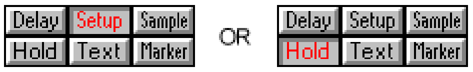
- Notice that the delays are added to the **Parameters in *group_name*** list box at the bottom of the dialog. To delete a parameter from this list, just select it and hit the **Delete** key on the keyboard.
- Normally the **Hide Default Clock Correlation Groups** check box is checked. Clocks automatically create 3 correlation groups to correlate the internal delays specified by the clock buffer delays (specified in *Edit Clock Parameters* dialog). The auto-generated groups will be named $$*Clockname*_BufferRising, $$*Clockname*_BufferFalling, and $$*Clockname* _BufferRisingFalling, but the user very rarely needs to edit these groups.

5.2 Setups and Holds

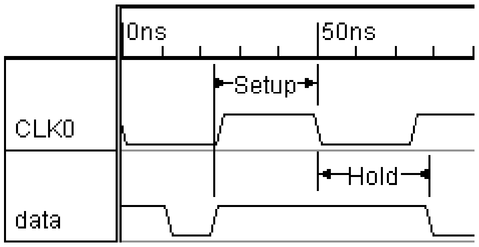
Setups are the minimum time necessary for a signal to be stable before a control signal transition, and **Holds** are the minimum time that a signal must be stable after a control signal transition. Setups and holds check timing constraint requirements for a design.

Add a Setup or Hold:

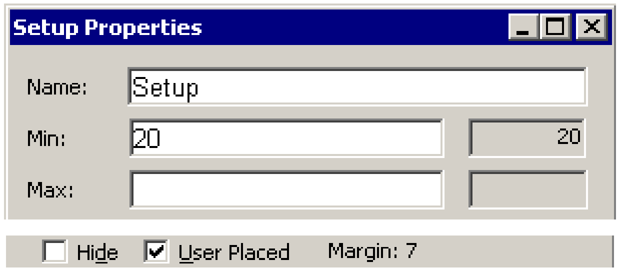
- Press the **Setup** or **Hold** button so that right clicks will add that type of parameter.



- Left click on the *data* signal to select it, then Right click on the *control* signal to add the parameter. To remember this order, remember that data signals have setup/hold constraints whereas control signals like clocks don't.



- If drawn correctly the arrows will point to the control signal. See Section 5.4 [83] on moving parameters after they are drawn.
- Double click on the parameter to open the *Properties* dialog and enter a **min** value. This is the minimum time that the data signal has to be stable before or after the control edge, and it can be a time value or time formula. The **max** value is rarely used, but if specified the data signal must occur between the min and max times. See Section 2.5 Time Formulas for Clocks and Parameters [47] for information on they syntax of the formulas.


Setup Properties

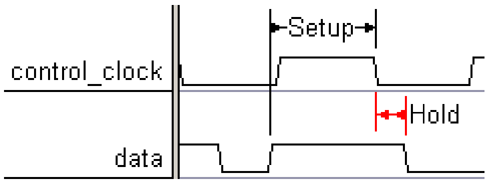
Name: Setup

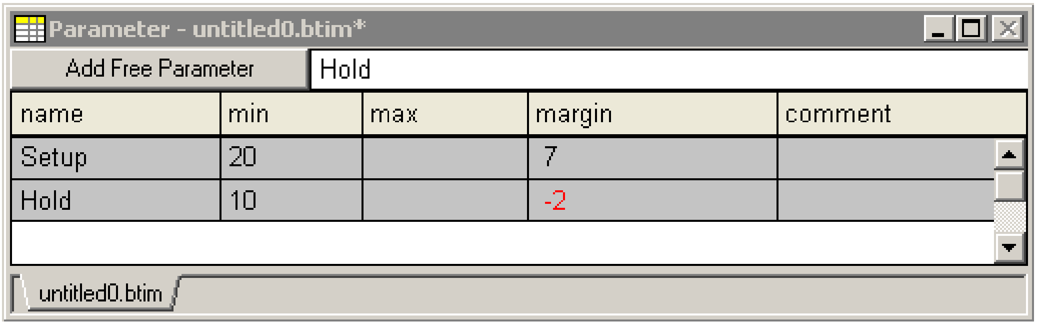
Min: 20 | 20

Max:

[ ] Hide [✓] User Placed Margin: 7

- **Margin** 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 *Diagram* window and the *Parameter* window.





Hold

name	min	max	margin	comment
Setup	20		7	
Hold	10		-2	

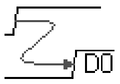
- The parameter can also be made to display the margin value by setting the **Display Label** to **min/max Margin** in the *Parameter Properties* dialog box.

5.3 Display Settings for Parameters

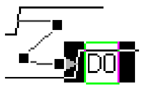
Delays, Setups, and Holds can be made to display as curved lines or with outward facing arrows. In addition to its name, a parameter can also display any of its' other properties or a combination of properties and text using a customized string. The user can also drag and drop the parameter to a new vertical position on the screen.

Curved Parameters:

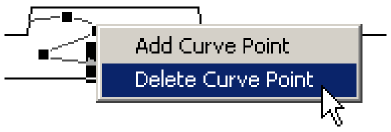
- Double click on a delay, setup, or hold parameter to open the *Properties* dialog and check the **Display as Curved Arrow** box.



- To edit the curve, select the parameter, and then drag and drop the black curve points. Or click and drag anywhere on the line to add another curve point.



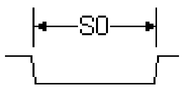
- To remove a curve point, right click on the curve point and choose **Delete Curve Point** from the context menu.



Tip: Adding a point very close to the end of a curved timing parameter can give you precise control over the exact angle of the arrowhead without changing the curve itself.

Setup and Holds with Outward Arrows:.

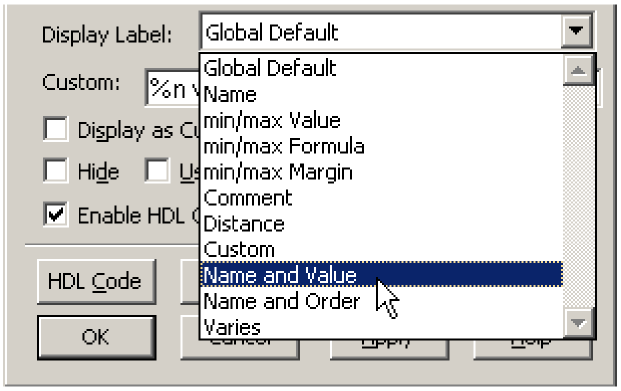
- Double click on parameter to open the *Properties* dialog, then check the **Outward Arrows** box.
- To do this globally, set it in **Options > Drawing Preferences** menu.



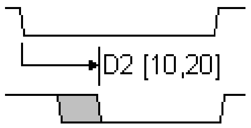
Displaying Different Information:.

A Parameter can display any of its properties or a customized string, rather than just its name.

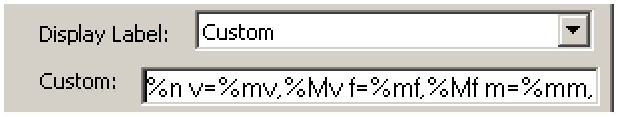
- Double click on a parameter to open the *Properties* dialog, pick one of the **Display Label** defaults. Note: that the *Distance* setting takes into account common delay removal effects, so it is the actual distance and not the visual distance.



- When this is applied, the parameter will display the new setting.



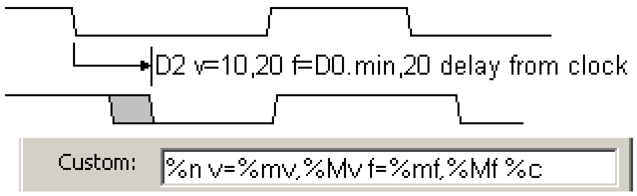
- If ***custom*** is selected in the Display Label box, then the parameter will display the string entered in to the **Custom** box. The control codes are replaced by the parameter values.



%n v=%mv,%Mv f=%mf,%Mf m=%mm,

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

- Tip: the default custom string has all of the control codes listed, so just start by editing that string.

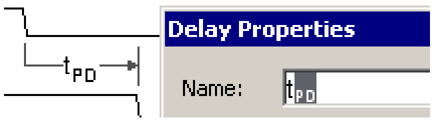


%n v=%mv,%Mv f=%mf,%Mf %c

- To change the display label for all parameters, select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog, and make the same changes there.

Formatting Parameter Names with superscript, subscript, bold, and italicize:

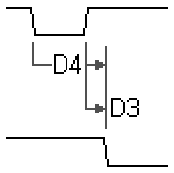
- Double click on a parameter to open the *Properties* dialog, then select the name text and press **Ctrl>-B** for bold, **<Ctrl>-I** for italize, **<Ctrl>-U** superscript (up), or **<Ctrl>-D** for subscript(down). Formatting is ignored during formula evaluation.



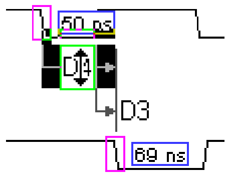
- To remove this formatting, just select the text and press the above formatting key combinations to toggle back to normal text.

Vertical Placement of Parameters:

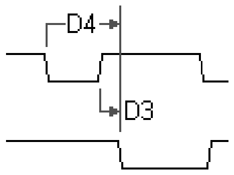
- Normally, the timing diagram editor attempts to show parameters so that all are visible at the particular zoom level.



- To change the position, click on the center of a parameter and drag it to a new vertical position. After you move a parameter it is considered *user placed* and the editor will not attempt to move it.

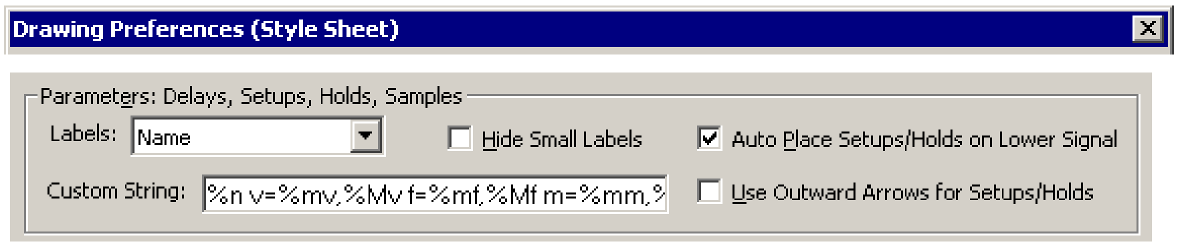


- To return placement control to the timing diagram editor, double-click on the parameter to open the *Parameter Properties* dialog, and uncheck the **user placed** box.



Change the default position of Setups and Holds:

- Choose **Options > Drawing Preferences** menu to open a dialog, and look that the **Parameters: Delays, Setups, Holds, Samples** section.


Drawing Preferences (Style Sheet)

Parameters: Delays, Setups, Holds, Samples

Labels: Name

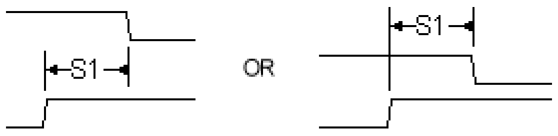
☐ Hide Small Labels

☑ Auto Place Setups/Holds on Lower Signal

Custom String: %n v=%mv,%Mv f=%mf,%Mf m=%mm,%

☐ Use Outward Arrows for Setups/Holds

- Check the **Auto place Setups/ Holds on lower signals** to make setups and holds appear above the lowest attached signal



Color Settings:

- To turn off the coloring of parameters that are shown in red, uncheck the **View > Show Bad Parms In Red** menu.
- To turn off the coloring of the parameters shown in blue and in green, uncheck the **View > Show Critical Paths** menu.
- To turn off the color background of unreferenced parameters, uncheck the **View > Show Unreferenced Parms in Gray** menu.

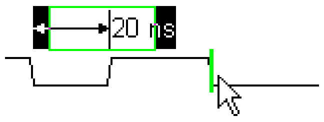
Hide Small Parameter Labels:

- Choose **Options > Drawing Preferences** menu, then check the **Hide small parameter labels** to allow parameter labels to 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.

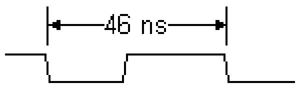
5.4 Drawing and Editing Parameters

Delays, Setups, Holds, and Samples parameters can be moved vertically on the screen. Also the edge attachments can be changed

Move to a parameter to a different edge:



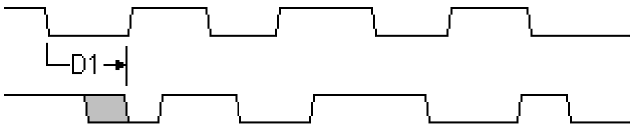
- Click on the parameter to select it. A selected parameter is surrounded by a rectangle with a solid handle box one or both ends.
- Drag one of the black handles to a new edge. The handle does not move but a green edge highlighter will jump to edge closest to the mouse. To abort the selection of a new edge, press the **<ESC>** key.
- Note: If the entire parameter is changing its vertical position 80 then you clicked on the middle of the parameter instead of a handle box.



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.

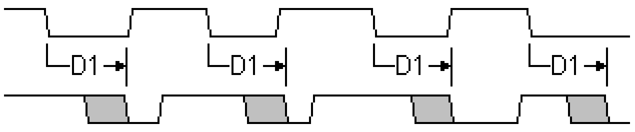
Repeat Parameters across the diagram:

- Add a delay, setup, or hold parameter to the diagram.



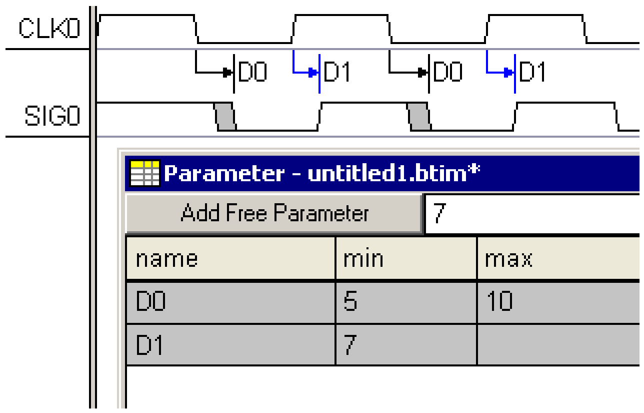
- Double click on the parameter to open the *Properties* dialog and press the **Repeat** button.

- The parameter will be repeated between like edges of the two signals. In the example the delay is added to the low to low edge.



Sharing Parameter Data:

- The editor assumes that a 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 (the same data line in the Parameter Window).
- Also naming a new parameter after an existing one will cause a sharing of parameter data.



7

name	min	max
D0	5	10
D1	7	

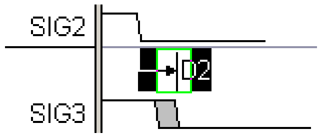
- To undo an automatic sharing, just rename the parameter in the diagram window to a new name.
- To turnoff parameter sharing globally, choose the **Options > General Preferences** menu to open a dialog, and uncheck the **Smart parameter sharing** box.
- Also, by default the program will display a warning message when you changed the name of a parameter to an existing parameter. Once you get use to the feature you can turn off this warning, by choosing the **Options > General Preferences** menu to open a dialog, and unchecking the **Warn if Duplicate Parameter Name** box.

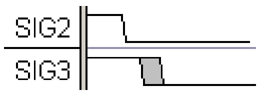
5.5 Hiding Parameters

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.

Hide selected parameters:

- Select the parameter and choose the **View > Hide Selected Parameter** menu. You can select multiple parameters by pushing the <CTRL> key while clicking the left mouse button.
- An alternate method is to check the **Hide** box in the *Properties* dialog.
- To show a hidden parameter, choose the **View > Show Hidden Parameter** menu option to open a dialog, then select the parameters to show from the list. If a parameter is attached to a hidden signal, it will continue to stay hidden until the signal is shown.





Hide all parameters on a selected signal:

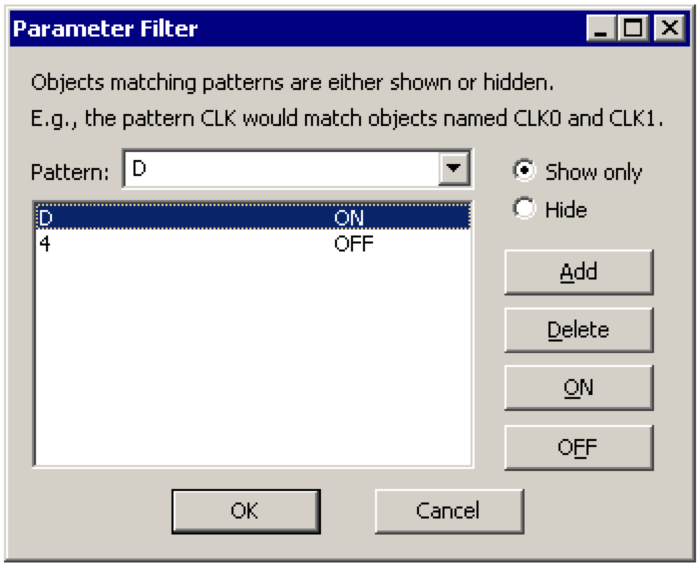
- Select one or more signal names, then right click and choose either **Hide Parameters on Selected Signal(s)** or **Show parameters on Selected Signal(s)** menu context menu.

Hide all parameters of a particular type:

- Selectively hide all delays, setups, holds, samples, or text by un-checking the **View > Show *parameter type*** menu.

Use Filters to Hide Parameters:

- Choose the **View > Filter Parameters** menu 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.
- 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. Turn patterns **on** and **off** using the those buttons.



Objects matching patterns are either shown or hidden.
E.g., the pattern CLK would match objects named CLK0 and CLK1.

Pattern: D

[✓ Show only] Hide

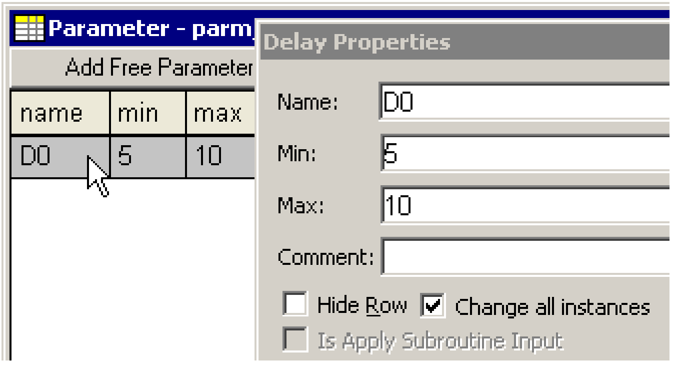
D	ON
4	OFF

5.6 Parameter Window

The Parameter Window shows parameters and free parameters in a row/column format.

Edit all instances of a parameter

- Double click on a parameter in the *Parameter* window to open the *Properties* dialog with the **Change all instances** box checked. Any changes, like to the **Display Label, Min,** or **Max,** will change every parameter of that name in the *Diagram* window.



name	min	max
D0	5	10

Name: D0

Min: 5

Max: 10

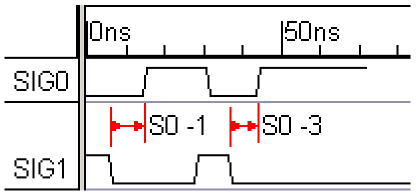
Comment:

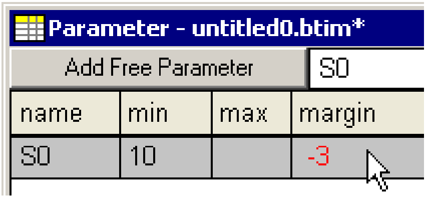
Hide Row [✓ Change all instances]

Is Apply Subroutine Input

Margin Display changes with selection in drawing window:

- When no instance of a setup or hold is selected in the *Drawing* window, the *Parameter* window displays the worst case margin for all the parameter instances.
- When an instance of a setup or hold is selected in the *Drawing* window, the *Parameter* window will show the margin (slack) for that instance.




Parameter - untitled0.btim*

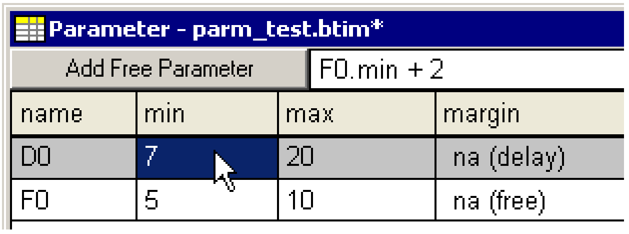
Add Free Parameter | S0

name	min	max	margin
S0	10		-3

Add a Free Parameter

A Free Parameter is a parameter that is not attached to any signal transitions in the *Diagram* window. They are used as variables in other parameters. They can also be grouped together and saved as libraries files, which is covered in Chapter 10: Libraries 138.

- Click the **Add Free Parameter** button in the *Parameter* window.
- Notice that F0 is in a white row to indicate that it is referenced by another parameter. To turn off the coloring uncheck the **View > Show Unreferenced Parms in Gray** menu.

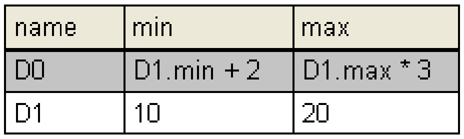

Parameter - parm_test.btim*

Add Free Parameter | F0.min + 2

name	min	max	margin
D0	7	20	na (delay)
F0	5	10	na (free)

Show Formulas instead of values

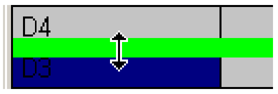
- Choose the **Options > Parameter Window Preferences** menu, and check the **Display min/max formula**, instead of the **Display min/max value**.



name	min	max
D0	D1.min + 2	D1.max * 3
D1	10	20

Move a Row

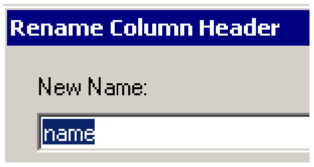
- Select the name of the row to be moved and place the cursor near the very top or bottom of the parameter so that the cursor changes to double arrow.



- Click and drag the mouse to the parameter's new location. Notice that as the mouse moves a green bar appears between the parameters.

Change Column Titles and Printing

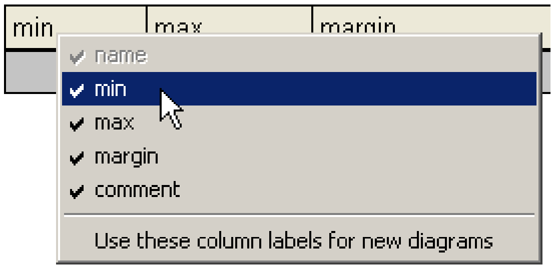
- To change the column title, double click on the column title to open a dialog for editing the name.


Rename Column Header

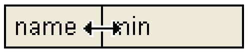
New Name:

name

- To remove columns from the Print image, right click on a column title and uncheck the the column to be excluded from printing.



- To adjust the width of a column, move the cursor to the line between the column titles. When the cursor changes to the double arrow, click and drag to a new position.



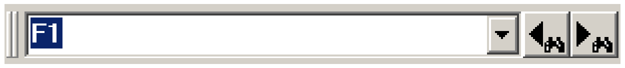
Hide a Parameter Row

Hiding a parameter in the *Parameter* window will hide its timing data and all instances of the parameter in the *Diagram* window.

- Either, select the parameter's name and choose the **View > Hide Selected Parameter Row** menu.
- Or, double-click to open the *Properties* dialog, and check the **Hide Row** box (notice this is different than the **Hide** box which only hides an instance of the parameter in the diagram window).
- To show a hidden parameter row, choose the **View > Show Hidden Parameter Row** menu.

Search for a Parameter name

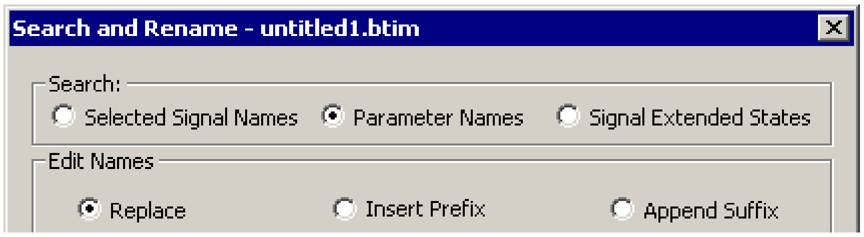
- To make the search box work on the *Parameter* window, click on the window so that it is the active window.
- Then type in a parameter name (or a partial name) into the Search box on the main menu bar.



- The forward search and backward search buttons will cause the Parameter window to scroll to the next match and highlight the parameter name.

Search and Rename Parameters

- To make the Search and Rename function work on the *Parameter* window, click on the window so that it is the active window.
- Choose the **Edit > Search and Rename** menu to open a dialog.



Search:
Selected Signal Names [✓ Parameter Names] Signal Extended States

Edit Names
[✓ Replace] Insert Prefix Append Suffix

- Select **Parameter Names** and the fill the rest of the fields with regular expressions that will match to partial and full names. See section 11.1 Signal Names for export and import 149 for examples of the regular expressions that can be used in the dialog.
- Sometimes during a rename operation the program can encounter a duplication conflict when a parameter is being renamed to the same name as another parameter, but that the two parameters differ in properties (different min and max times) or types (delay, setup, hold). If such a conflict is found the *Duplicated Parameter Names* dialog will open so that you can fix the conflict. This dialog is described in Section 10.4 Merging Diagrams 144, because this type of conflict happens frequently during a timing diagram merge.

5.7 Time Formulas for Clocks and Parameters

The information below is a duplicate of the material in section 2.5 of the clocks chapter, but it's repeated here because of it's high importance for specifying timing parameters.

A Time formula can be used wherever a time value is required. For example the min/max for parameters, clock period, clock offset, and clock jitter all can accept time formulas. Time formulas are in display time units (see 1.10 Base and Display Time Units 36). Some examples of time formulas:

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

- A quick way to grab a parameter name for use in a formula is to click into the min or max box in the *Properties* dialog, then press the **Library** button to open the *View Parameters in Libraries* dialog (see section 10.2 Referencing Parameters 140). This will list all parameters in the project and the associated libraries, once you select a parameter it will be inserted into the box in the *Properties* dialog that you were in.

Time formulas are algebraic combinations of the following:

- **Parameter names**, **parameters with dot attributes**, and **clocks with dot attributes** can be used in formulas. If just the parameter name is used, 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.

```
D0
D1.min
D2.max
Clk0.period
```

- **Function Calls** defined in the Parameter window. The dot-max and the dot-min properties can be used to force the equation to use a particular value. Also, the input arguments can be used in the expressions.

```
f(4,5)
func.max(2+D0,4)
```

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

```
5
2.010
255E-3
```

- **Constants** are fixed point numbers that are *unit-less*. 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*

```
'2
'4.1e2
```

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.

- **Macro names** enclosed with percentage marks will be replace with the macro value as defined in the *Edit Formula Macro* dialog. See Section 10.1 Adding Libraries, Specifications, and Macros 138 .

```
%lib_spec$.F0
```

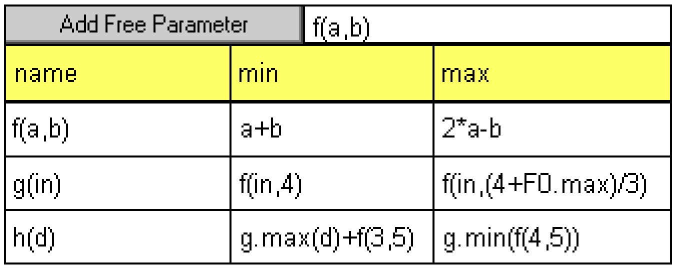
Dot Attributes

- *Parameters* and *User defined functions* support **min** and **max** attributes.
- *Clocks* can only be referenced with an attribute of **period**, **duty**, **offset**, and **jrise** (rising jitter) and **jfall** (falling jitter), and the four buffer delays: **minLH**, **maxLH**, **minHL**, **maxHL**.

```
F(5).min
D0.max
Clk.duty
Clk.offset
Clk.jrise
```

Function Definitions in the Parameter Window

- Click the **Add Free Parameter** button in the *Parameter* window, to add a free parameter.
- Double click on the free parameter to open the *Properties* dialog.



name	min	max
f(a,b)	a+b	2*a-b
g(in)	f(in,4)	f(in,(4+F0.max)/3)
h(d)	g.max(d)+f(3,5)	g.min(f(4,5))

- The function name defines both the name and the input argument list. There must be a least one argument. Arguments are separated by commas.

```
FunctionName(input1<,Input2,….,inputN>)
```

- The min and max fields define the body of the function.

Built in Functions supported:

Function	**Description**
abs	Returns the absolute value of the number.
acos	Returns the arccosine of a number in radians. The number must be a value from –1 to 1.
asin	Returns the arcsine of a number in radians. The number must be a value from –1 to 1.
atan	Returns the arctangent of a number in radians.
cos	Returns the cosine of a number expressed in radians.
cosh	Returns the hyperbolic cosine of the number.
degrees	Converts radians to degrees.

exp	Returns e raised to the power of the number.
fact	Returns the factorial of the number. The number must be in the range of 0 to 69. If the number is not an integer, the decimal will be truncated.
int	Rounds the number down to the nearest integer. `Example: int 4.1 = 4`
ln	Returns the natural logarithm (log base e) of the number. The number must be greater than zero.
log	Returns the logarithm to the base 10 of the number. The number must be greater than zero.
radians	Converts degrees to radians.
rand	Returns a random integer between 0 and 32,767 based on the seed number given.
sign	Returns -1 if the number is negative, 1 if the number is positive, and zero if the number is zero.
sin	Returns the sine of the number expressed in radians.
sinh	Returns the hyperbolic sine of the number.
sqr	Returns the square root of the number.
sqrt	Returns the square root of the number.
tan	Returns the tangent of the number expressed in radians.
tanh	Returns the hyperbolic tangent of the number.
trunc	Truncates the number to an integer by removing the decimal. `Example: trunc 4.1 = 4`

Mathematical Operators

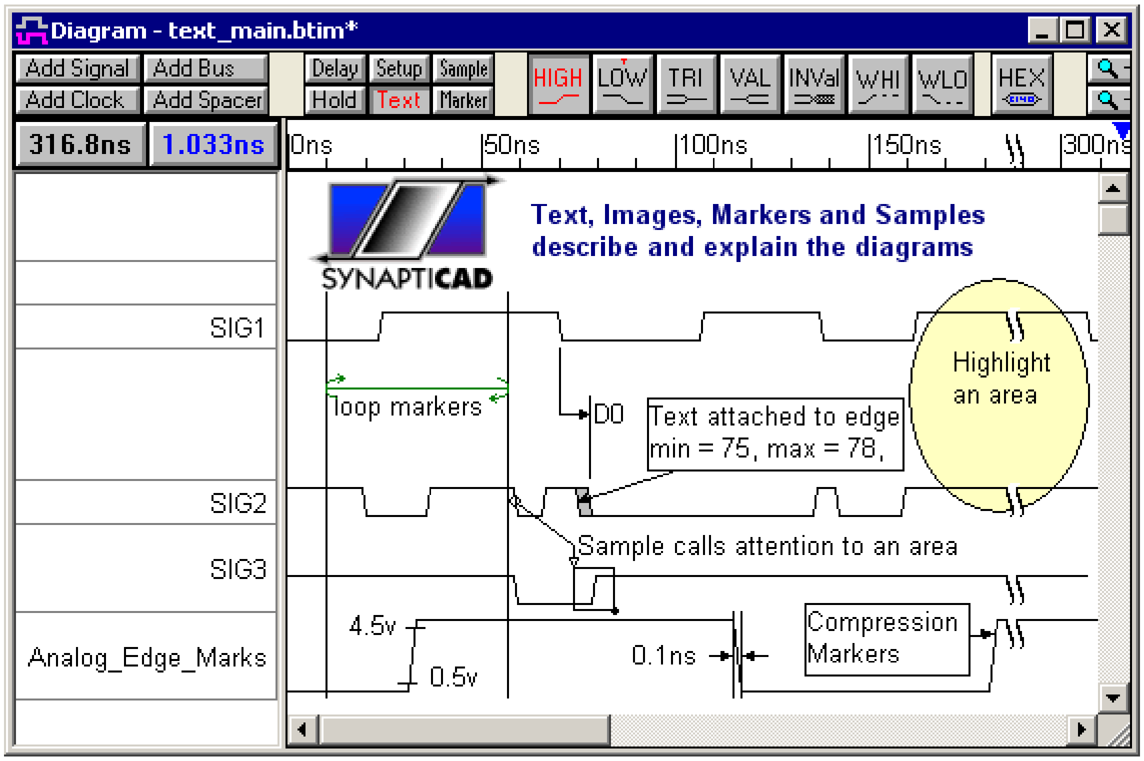
The timing diagram editor supports the **+**, **-**, *****, and **/** mathematical operators. The editor also uses the operator ^ to raise a number to a power.

No Circular Dependencies:

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.

Chapter 6: Text, Markers, and Samples

Text, Marker lines, and Samples can be used to annotate the timing diagram. Special events can be called to attention by attaching a text to the edge. Special States can have a sample describing the activity. Text objects can be placed anywhere with complete font, size, and color control. And in DataSheet Pro images (like logos) can be imbedded directly into the timing diagram.

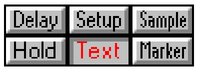


6.1 Text

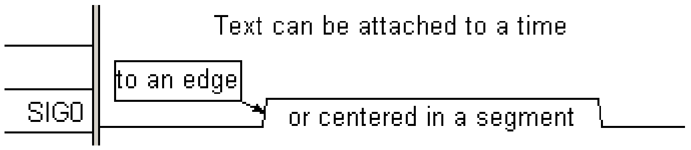
Text objects can be quickly added to a timing diagram by using the Text Mode button. Further editing allows font, color, formatting, and the attachment to be change.

Add a simple text object:

- Press the **Text** button so that right clicks will add text objects.

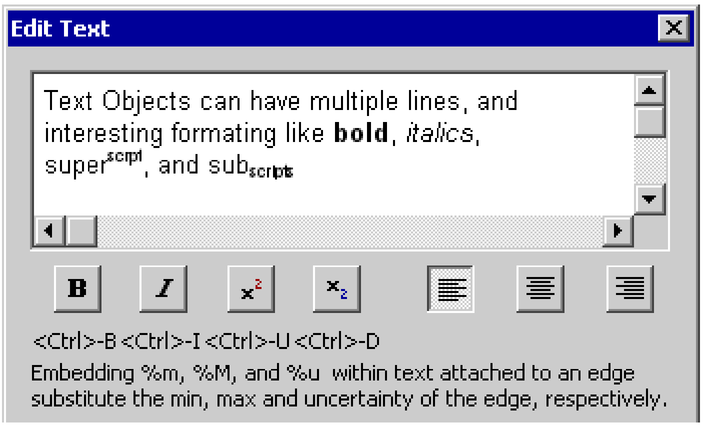


- Right click to open and edit box and type in the text. When text is added it will be automatically attached to a preselected edge, a preselected segment, or attached to time (if no edge or segment was selected before the text object was added).



Edit the text object:

- Double click on the text object to open the *Edit Text* dialog.
- Multiple lines can be added by using the <Enter> key.
- Select some of the text and then press the bold, italic, superscript, or subscript styles to format the text.
- The left, right, and center justification affects the text in the diagram but not inside the dialog itself.

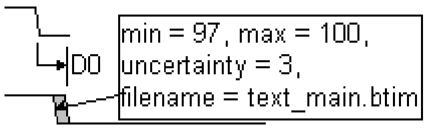


Text Objects can have multiple lines, and interesting formating like **bold**, *italics*, super^{script}, and sub$_{scripts}$

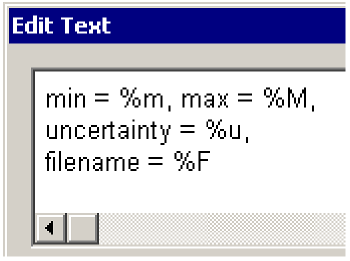
<Ctrl>-B <Ctrl>-I <Ctrl>-U <Ctrl>-D

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

- If text is attached to an edge, the min, max and uncertainty of the edge can be displayed using the dynamic substitution codes %m, %M, and %u.
- Also the name of the timing diagram file can be displayed with and without the file extension using the dynamic substitution codes %F and %f. The file name control code can be displayed in any text object.



min = 97, max = 100, uncertainty = 3, filename = text_main.btim

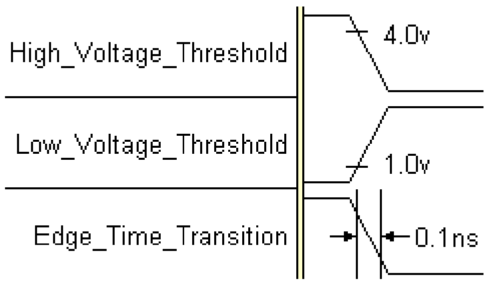


min = %m, max = %M, uncertainty = %u, filename = %F

- To change the font or color of the text, press the **Font** button to open the *Font* dialog.
- The default font settings are set at **Options > Text/Color Preferences > Set Diagram Window Font**.

Font...
Font and Color Control

- Edge Threshold tick marks and values are displayed using text object. See Section 8.4 Display Edge Switching Thresholds[123] for more information

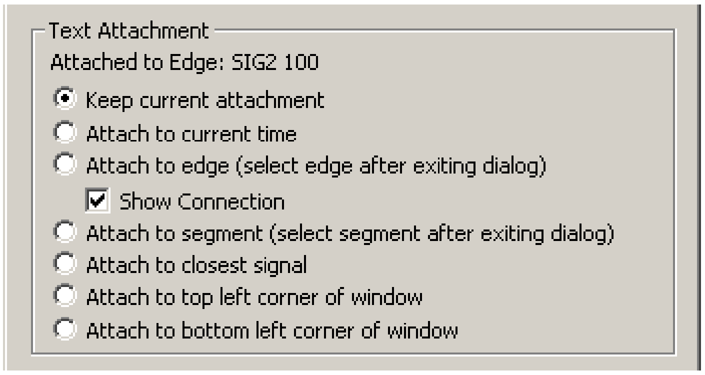


Text Attachment controls when and how text is displayed

When a simple text object is added, it is automatically attached to both a signal and either a time, event, or segment. This automatic attachment can be changed using the *Edit Text* dialog. There are also special attachments that fix the text at the top or bottom of a timing diagram.

- Inside the *Edit Text* dialog, the **attached to *something: at some place*** shows the current attachment.

- Select the radio button for the new attachment and then click **OK** to close the dialog.



Text Attachment
Attached to Edge: SIG2 100
[● Keep current attachment]
○ Attach to current time
○ Attach to edge (select edge after exiting dialog)
☑ Show Connection
○ Attach to segment (select segment after exiting dialog)
○ Attach to closest signal
○ Attach to top left corner of window
○ Attach to bottom left corner of window

- If you chose **Attach to edge**, **Attach to segment**, or **Attach to closest signal**, when you close this dialog the timing diagram editor go into a continuous selection mode. After closing the dialog, immediately move the mouse to the desired signal edge or segment and click on it to attach the text the element. To abort this operation, press the **<ESC>** key.

- If you chose **Attach to current time**, then the text will be fixed at that time.

- Checking **Show Connection** causes a box to be drawn around the text object and an arrow that points to the attached edge.

- If you chose **Attach to top left corner of window** or **Attach to bottom left corner of window**, the text will be fixed in a specific position of the window. Regardless of how the diagram is scrolled or zoomed the text will be fixed in the specific position on the screen. This is handy for titles and logos.

Hidden Signals and Text Objects:

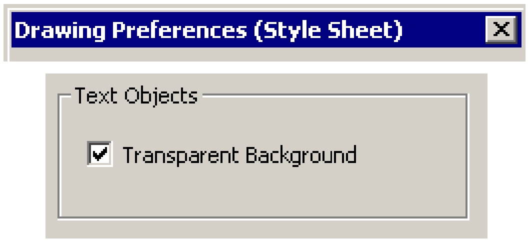
Normally, hiding a signal will also hide the text and grid lines attached to the signal. However, sometimes it may convenient to use a signal just for holding text or grid lines, but otherwise that signal is unrelated to your circuit design and does not need to be displayed. The checking **View > Show Hidden Text** menu, then hiding the documentation signal will allow the text to continue to be displayed even though the signal is hidden.

Transparent or Opaque boxes Behind the Text:

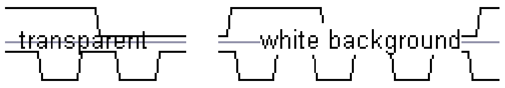
By default all text objects have a transparent background so that you can see the waveforms and parameters that appear under the text. This is especially handy for when you are working at different zoom levels. However, for printing and image generation you may wish to place a white background to all of the text objects.

- Choose the **Options > Drawing Preferences** menu to open the preferences dialog.

- Uncheck the **Transparent Background** check box to create an opaque white box behind the text.



Drawing Preferences (Style Sheet)
Text Objects
☑ Transparent Background



6.2 Moving Text on the Alignment Grid

Text can be moved using the mouse or the keyboard. Text that 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. When text is moved it automatically snaps to the closest text alignment grid intersection.

Drag-and-drop text with the mouse:

- Click on the text object and drag it to the desired location.

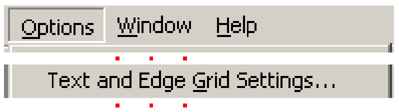
Nudge the text using the keyboard:

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

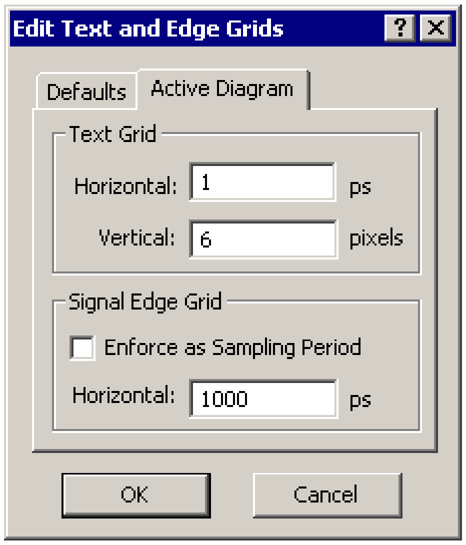
Change the text alignment grid:

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

- Choose the **Options > Text and Edge Grid Settings** menu item to open the *Edit Text and Edge Grids* dialog.



- In the **Text Grid** section, enter the new ***horizontal*** and ***vertical*** values. By default, we set the Vertical limit to a rather large 6 pixels to make it easy to line up text in a vertical direction, and use a very small value for the horizontal direction to give the most freedom of placement.
- The **Active Diagram** and **Defaults** tabs, determine whether the settings affect the current active diagram or whether the settings will be used as the defaults for new diagrams.



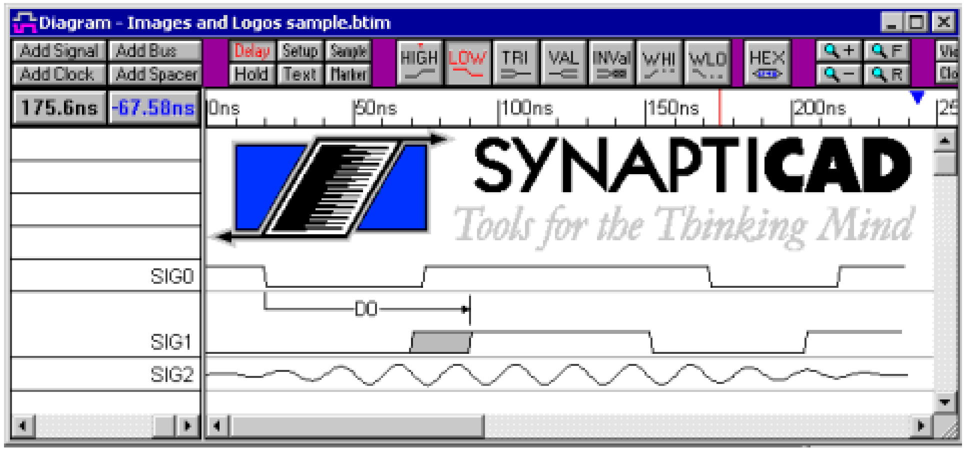
Text Grid
Horizontal: 1 ps
Vertical: 6 pixels
Signal Edge Grid
☐ Enforce as Sampling Period
Horizontal: 1000 ps

- 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.

6.3 Image and Logo Text Objects

DataSheet Pro allows images and company logo files to be embedded into a timing diagram. Once the image is saved by DataSheel Pro, it can be viewed in any of SynaptiCAD's timing diagram editor products including the free WaveViewer product.

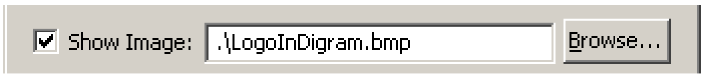
The link to the image file is stored in a Text Object and has the same kind of attributes as a normal text object that allow the image to be locked to a particular place on the screen or attached to a particular edge or signal. Only a relative path to the image file is stored in the btim file, so when the timing diagram is distributed you must also distribute any necessary image files.



To display a logo (or any image) in a timing diagram:

- Create a text object and type in some text. This text will be text for the tool-tip fly-out that pops up when the user puts his mouse over the image.

- Double click on the text object to open the *Edit Text* dialog, and check the **Show Image** box.



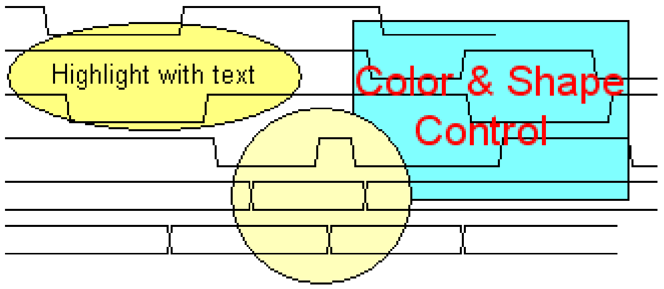
☑ Show Image: .\LogoInDigram.bmp Browse...

- Specify an ***image file*** to display by either typing in a file name, or clicking the **Browse** button to select a file. The following image formats are supported:
  - Bitmap (*.bmp)
  - CompuServe Graphics Interchange Format (*.gif)
  - Icon Files (*.ico)
  - JPEG Images (*.jpg, *.jpeg, *.jif, *.jfif)
  - Portable Network Graphics (*.png)

- When the *Edit Text* dialog is closed the image will appear on the timing diagram.

6.4 Highlight Areas with Text Objects

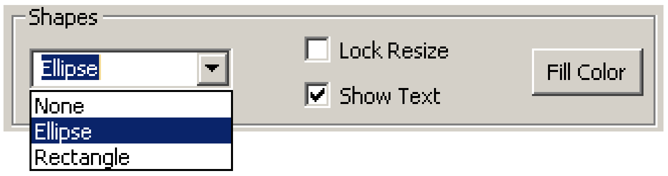
Text Objects can be used to highlight an area on the timing diagram.



Adding a Highlight Region:

- Create a text object and type in some text as shown in Section 6.1 Text 91.

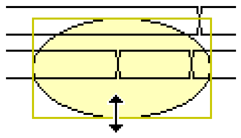
- Double click on the text object to open the *Edit Text* dialog, and pick a shape from the **Shapes** box.
- **Show Text** determines whether the text will be shown or not.



- **Fill Color** determines the background color of the the shape.
- **Lock Resize** will fix the size of the area and not allow any mouse editing.
- Press OK to close the dialog and have the area appear.

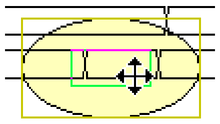
Resizing the Highlight region:

- Place the Mouse over the edge of the region until the the cursor changes shape, then drag and drop the edge to the new size.



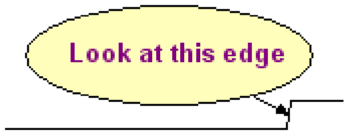
Moving a Highlighted Region:

- Left mouse click near the center of the highlighted region so that a green selection box appears in the center.
- Place the mouse over the green selection and drag and drop to a new location.
- Also Up, Down, Left, and Right arrows can be used to nudge the text to a new position.



Highlights with arrows and other fonts:

- Areas can point to an edge, by setting the **Attach to Edge** control in the Edit Text Dialog.
- Also the font of the text can be controlled by the **Font** button at the bottom of the dialog.

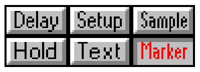


6.5 Marker Lines

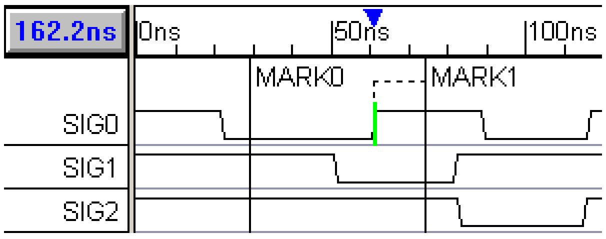
Markers are vertical lines that display their placement time in the timing diagram. Once a marker is added to the timing diagram, its behavior can be altered so that it compresses time, mark the end of the diagram, create a loop in the diagram, or call an HDL code segments.

To add a time marker:

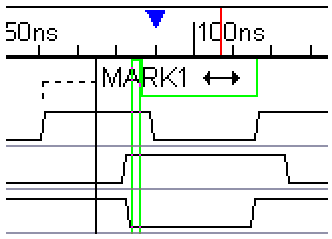
- Press the **Marker** button so that right clicks will add marker lines to the diagram.



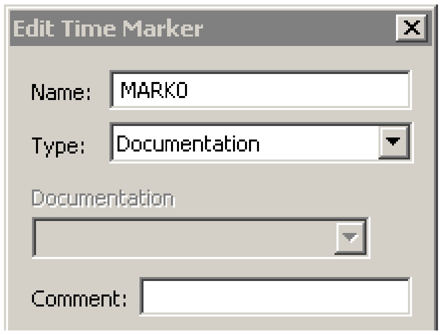
- To attach a marker to a ***time***, make sure no edges are selected, then right click on a waveform to add a marker.
- To attach a marker to an ***edge***, select the control edge, then right click. If the edge is moved, the the marker will stay a fixed distance from the edge



- To move a time marker, drag and drop the time marker line or name.



- To edit a marker, double click on the marker to open the *Edit Time Marker* dialog.
- The **name** field for markers can be blank, so that by default the maker only displays its line. This is the only object besides *spacer signals* that can be nameless.


Edit Time Marker

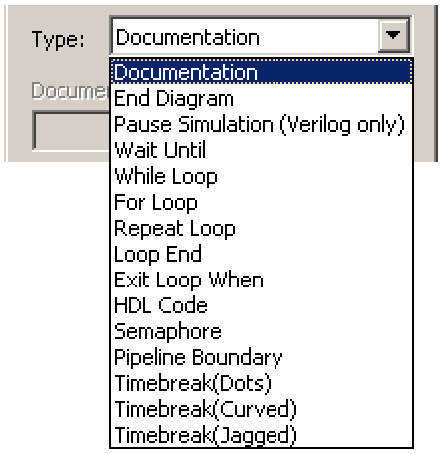
Name: MARK0

Type: Documentation

Documentation

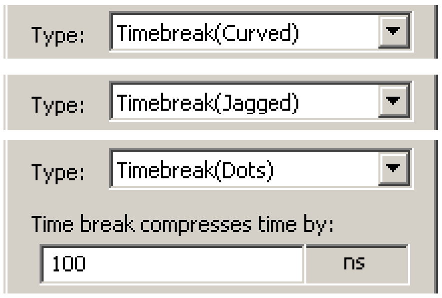
Comment:

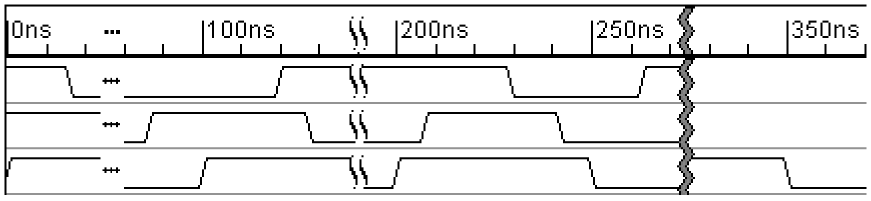
- The **Type** determines the behavior of the marker.
- **Documentation** is a marker with a solid line that can also display time.
- **End Diagram** stops stimulus generation at that point.
- **Pause Simulation**, **Wait Until**, **Exit Loop When**, **HDL Code**, **Semaphore**, and **Pipeline Boundary** are TestBencher Pro and Reactive Test Bench features, but in the timing diagram editors they behave like **Documentation** markers. See the Marker Chapters in those manuals.



Time Compression Markers:

- Choose one of the **Timebreak** types to make a dotted, curved, or jagged line.
- To compress time to the right of the marker, ender a time into the **time break compress time by** box. The compressed time still exists, but it does not display on the screen.


Type: Timebreak(Curved)
Type: Timebreak(Jagged)
Type: Timebreak(Dots)
Time break compresses time by: 100 ns

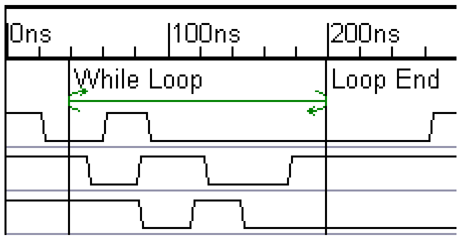


Dotted timebreak Curved timebreak Jagged timebreak

Loop Markers:

Loop Markers are pairs of markers that draw a line between themselves to show that part of the timing diagram should be repeated. If you are adding loops for test bench generation, please read the marker chapters in the TestBencher Pro or Reactive Test Bench Generation Manual. This section is only about making pretty looking markers.

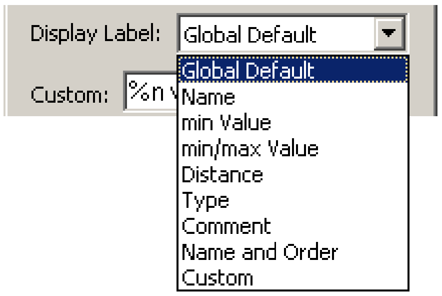
- Add two markers to the diagram.
- For the left most marker choose one of the loop start types of **While Loop**, **For Loop**, or **Repeat Loop**.
- For the right most marker choose the **Loop End** type.



- To make this picture we also set the **Display Label** to **Type** so that the markers would show their types.

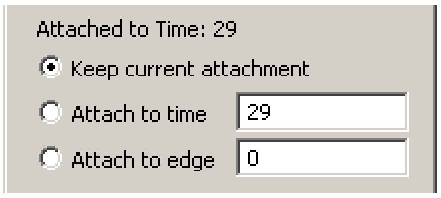
Display Name or any other property:

- Markers can display any of their properties listed in the **Display Label** box. The global default is set in the **Options > Drawing Preferences** menu.
- The **Custom** label displays the string in the **Custom** box with the control codes replace by values. For an example of customs display strings, see ***Displaying Different Information*** for parameters in Section 5.3: Display Settings for Parameters 80.
- The **Type** label for documentation markers is Blank so that you can just have a line and no other label.


Display Label: Global Default
Custom: %n \

```
name = %n
min value = %mv
max value = %Mv
distance = %md
type = %t
```

Change the Attachment of the Marker:



Attached to Time: 29

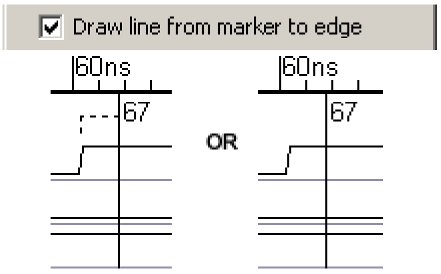
[✓ Keep current attachment]

Attach to time 29

Attach to edge 0

- The top of the box lists either the time or the signal and time of the current attachment.
- Choose **Attach to time** and type in the new time. The display label for these types of markers will float at the top of the screen so as you scroll vertically through the diagram you can still see the marker display label.
- Choose **Attach to edge** AND close the dialog. The editor will be in a continuous selection mode with a green edge highlight moving to different edges. Click on the new edge to set the attachment. Note the box to the right is the distance of the marker to the edge.

Edge lines for Markers:



☑ Draw line from marker to edge

- If the marker is attached to an edge, the **Draw line from marker to edge** box controls whether the attach to edge dotted line will be displayed

Snap Ends to Marker

☑ Signal ends snap to marker

- Check the **Signal ends snap to marker** box and 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. See Section 1.2: Drawing Waveforms 14 for an extend example under the "Making Waveforms end at a common time".

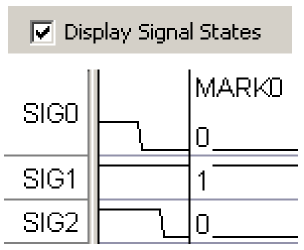
Display Label Position

☑ Auto-Adjust Display Label Position

- The Marker Label can be dragged and dropped so that it appears between two signals. Once this is done you can check the **Auto-Adjust Display Label Position** to return label placement control to the editor.

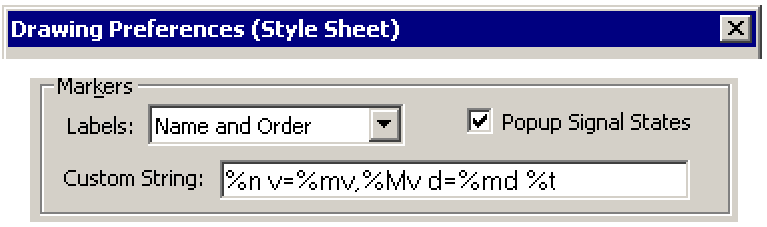
Display Signal States with the Marker

- Markers can permanently display the signal state values if the **Display Signal States** box is checked in the *Marker Properties* dialog. Otherwise the signal states are just shown when the mouse hovers over the marker line.



Global Settings for Markers

- Select the **Options > Drawing Preferences** menu to open the *Drawing Preferences* dialog and look at the **Markers** section.
- **Popup Signal States** controls whether markers show their states when the mouse sweeps over them.
- **Labels** and **Custom String** are the default settings for the same controls in the *Marker Properties* dialog.

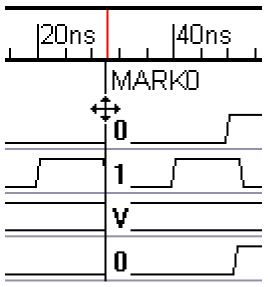

Drawing Preferences (Style Sheet)

Markers

Labels: Name and Order

[✓] Popup Signal States

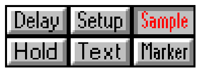
Custom String: %n v=%mv,%Mv d=%md %t

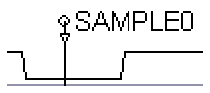


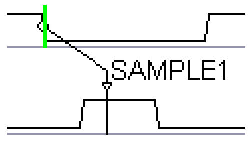
6.6 Samples

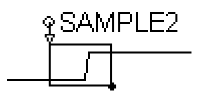
For the timing diagram editors, samples are just graphical constructs that do not have any effect on the timing of the waveforms. However, samples are used by TestBencher Pro and Reactive Test Bench Generation to indicate points that are to be watched during simulation. If you are adding samples for test bench generation, please read the sample chapters in those manuals. This section is only about making pretty looking samples.

- Press the **Sample** button so that right clicks will add samples to the diagram.
- To attach a sample to a time, make sure no edges are selected, then right click on a waveform to add a sample.
- To attach a sample to an edge, select the control edge, then right click on the waveform to be sampled.
- To make a box sample, double click on a sample to open the *Sample Properties* dialog, and set the **min** and **max** values to define the box.

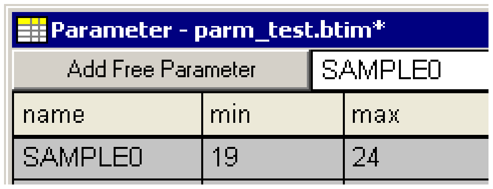








- Samples are also added to the Parameter Properties dialog.


Parameter - parm_test.btim*

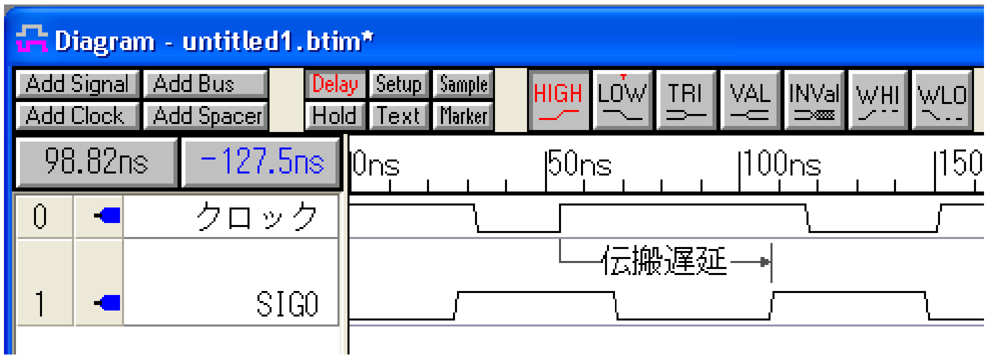
name	min	max
SAMPLE0	19	24

Change how the Sample looks:

Samples are parameters (just like delays, setups, and holds) so most all of the parameter editing techniques also work for samples. See 5.3 Display Settings for Parameters 80, 5.4 Drawing and Editing Parameters 83, 5.5 Hiding Parameters 84, and 5.6 Parameter Window 85.

6.7 Japanese and other non-English Fonts

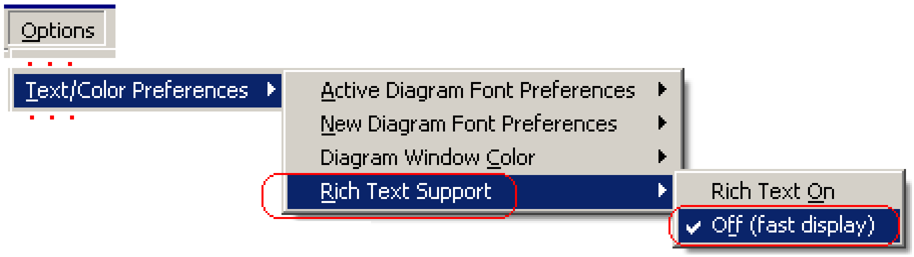
The timing diagram editor can support Japanese and other non-English fonts. This support requires that you disable the Rich Text Formatting and Set the Font types for the various windows to the non-English font type.



Disable Rich Text Formatting:

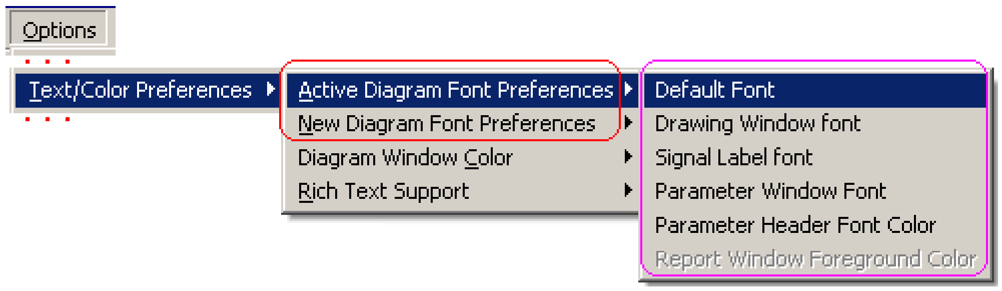
SynaptiCAD's timing diagram editors by default use Rich Text Format (RTF) display that supports superscripting and subscripting of English Characters. For non-English character sets to function properly, you will need to disable the RTF display (disabling superscripting and subscripting).

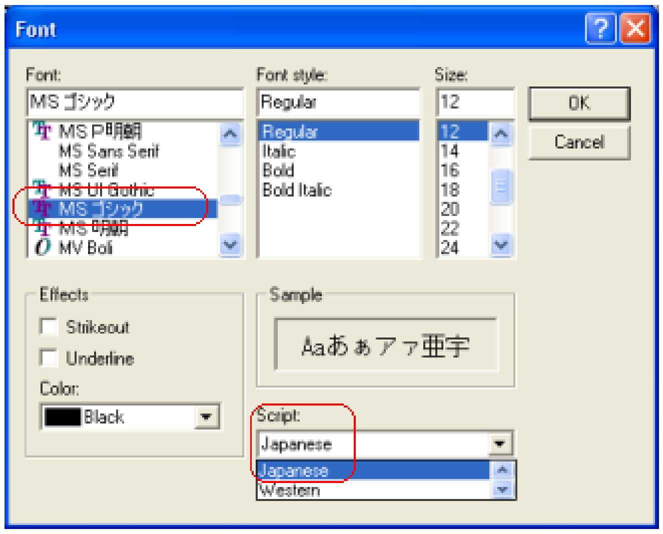
- Check the menu **Options > Text/Color Preferences > Rich Text Support > Off (fast display)** menu.



Set Font Types:

- Under the **Options > Text/Color Preferences** menu, use either the **Active Diagram Font Preferences** menus to set the fonts for the *current diagram*, or **New Diagram Font Preferences** menus to set the font for *all new diagrams* that you will create in the future.



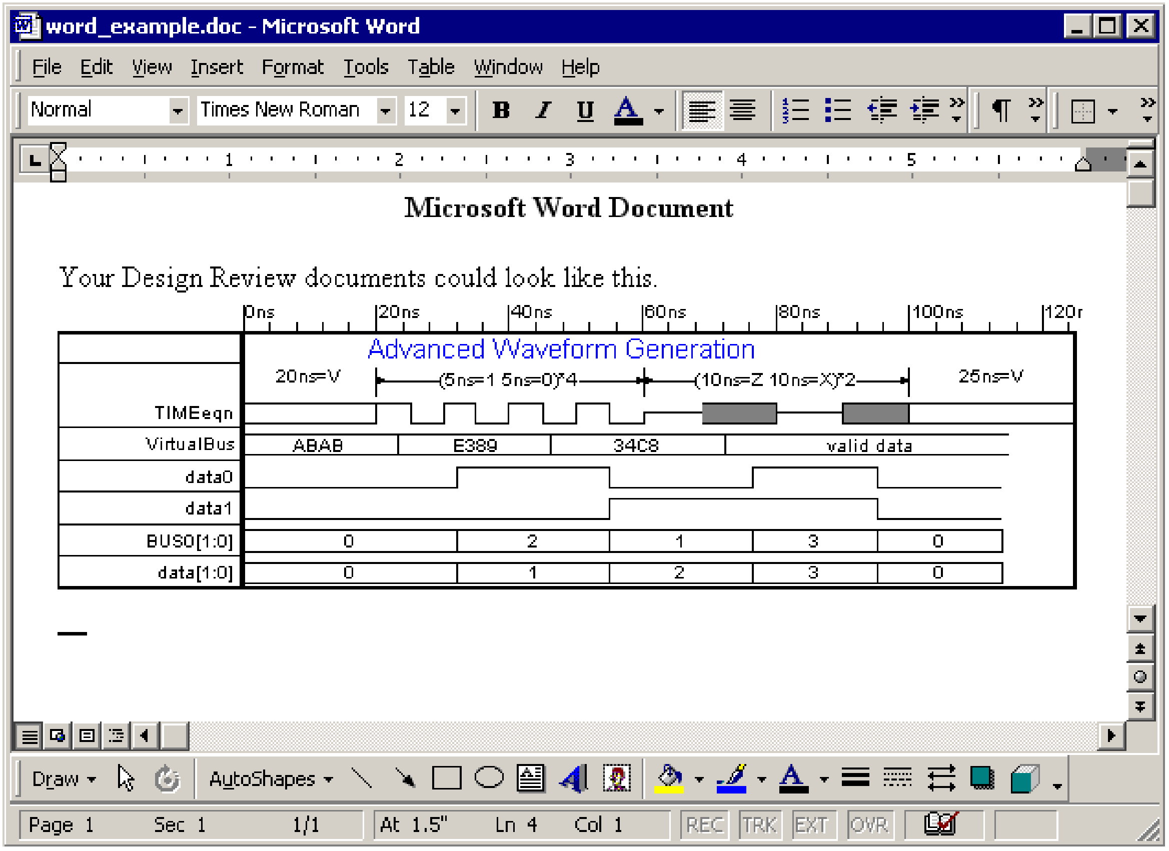


- Choosing one of the above **font** menus will open the *Font* dialog.
- Set the **Font** type to the non-English font that you want. For Japanese, it might be a font like MSMincho or MSGothic.
- Set the **Script** to the language that you want (e.g. instead of ***Western***, use ***Japanese***).
- This step needs to be done for the signal label window, drawing window, and parameter window fonts (and probably the default font as well) in order for Japanese characters to render properly in these windows.

Note that currently the *Parameter Properties* dialog will still display the names garbled, but the parameters will display properly in the diagram window and on printouts and diagram images. Non-English fonts are not supported in the equation box, so you cannot use an equation like (D0.min+2) in another delay's min box if D0 is in a non-English character set.

Chapter 7: Views, Images, and Printing

This chapter discusses features for first getting the timing diagram to display the relevant sections by using views, zoom, and scrolling features. Then it discusses the different was to get those sections into your design documents using image files, OLE, and printing features. It also discusses several features that are specific to DataSheet Pro to handle large numbers of images and timing diagrams.

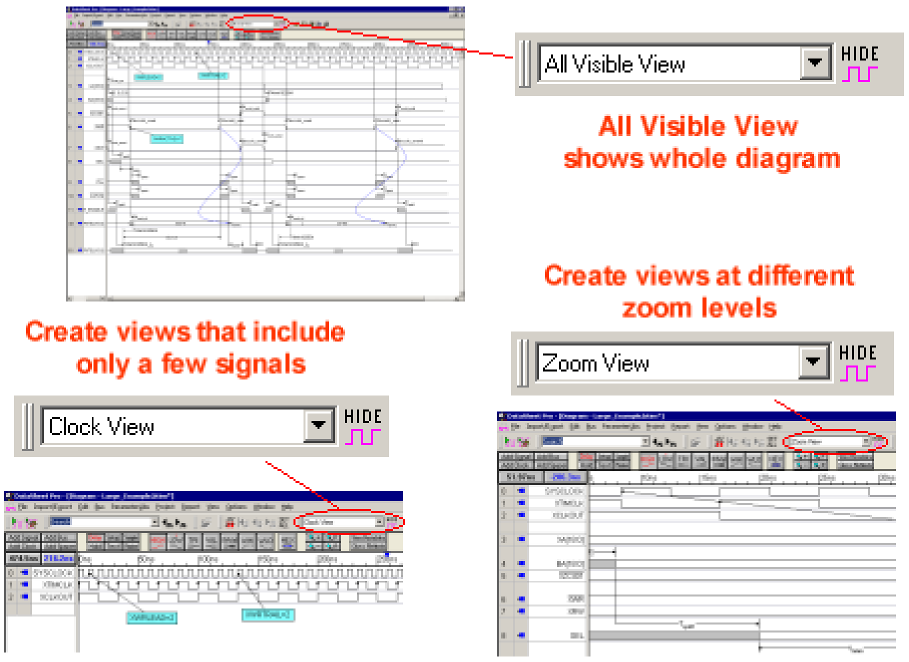

word_example.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
12
Draw
AutoShapes
Page 1 Sec 1 1/1 At 1.5" Ln 4 Col 1 REC TRK EXT OVR

Microsoft Word Document

Your Design Review documents could look like this.

7.1 Views in DataSheet Pro

DataSheet Pro users can create Views of the diagram, then use the Views to jump to particular sections of the diagram or zoom levels. This feature captures the current zoom and scroll levels and automatically displays that position in the diagram. Views can also be used to create different images of the same timing diagram.


All Visible View
HIDE
All Visible View shows whole diagram
Create views at different zoom levels
Zoom View
HIDE
Create views that include only a few signals
Clock View
HIDE

Create a DataSheet Pro View for quick viewing or printing specific sections:

- Adjust the timing diagram so that it shows only what you want to see in the view that you are creating. You can hide different signals and parameters in each View, so a View can radically change the information displayed in the window (See Section 1.9 Hiding Signals 34).
- Select the **View > Image View** menu to open the *Image View Capture* dialog.
- Press the **New** button and enter the name of the view in the dialog that pops up. This creates a view with what is currently visible in the diagram.

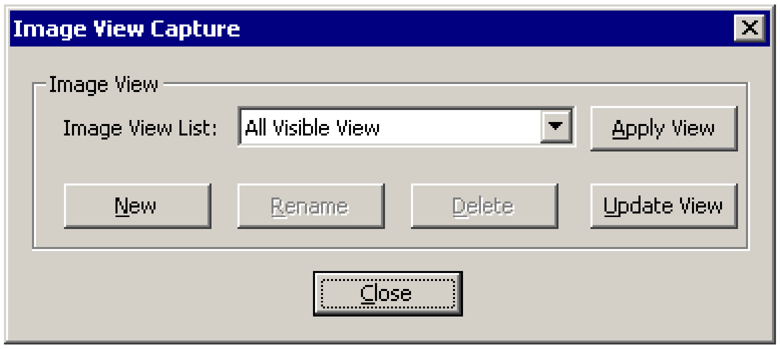
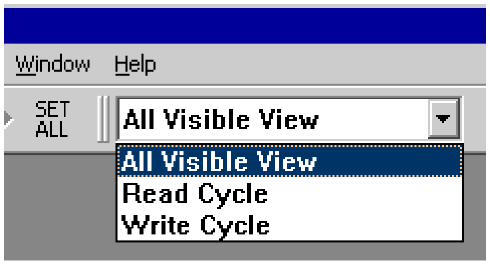


Image View List: All Visible View

- Once a View is created, the **Update View** button will update the view such that what is currently shown on the screen will now be in the named view.
- You can **Delete** a view or **Rename** it using the those buttons.
- The **Apply View** button changes the diagram to display that particular view.

- The fastest way to switch between views is to choose a View from the drop-down box on the tool bar. This will automatically change the diagram to display the new view.
- The **All Visible View** is the default view for the program and displays the entire timing diagram.



- **After a View is created**, any changes to the diagram (like new signals and text) will change all of the existing views.

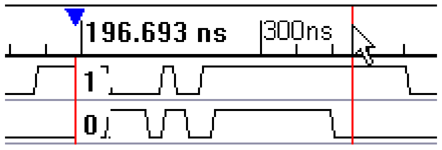
7.2 Zoom, Scroll, Search

Besides the normal scrolling and zooming functions, the timing diagram editor can quickly scroll to a specific time or signal, and zoom over a specific range.

Zoom using buttons or the mouse

- To zoom in and out quickly, hold down the **<Shift>** key while using the **scroll wheel** on your mouse.
- To zoom in over a visible section, drag and drop inside the **Time Line.**

<Shift> and mouse **scroll wheel**



- The zoom buttons on the diagram window button bar and in the **View** menu.
- The zoom in (+) and zoom out (-) center the zoom on the selected item, the blue delta mark, or the center of the diagram in that order.

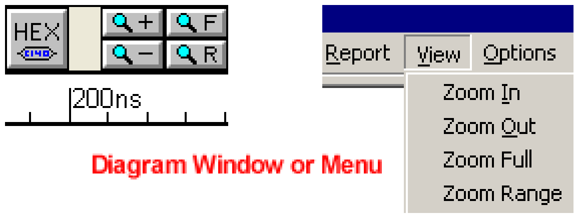
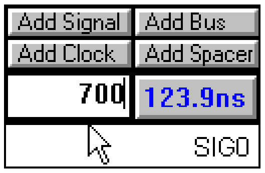


Diagram Window or Menu

- The zoom full (F) displays the entire timing diagram on the screen.
- The zoom range (R) opens a dialog that lets you specify the starting and ending times for the zoom.

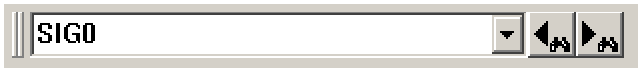
Scroll to a specific or relative time using the Time or Delta buttons:

- Press on either the **Time** or the **Delta** button to open an edit box, and type in a time. The Time button (black) causes the diagram window to scroll to that exact time. The Delta button (blue) causes the diagram window to scroll that amount from its current position.



Search for a specific signal name, parameter name or string:

- Select one of the child windows in the program, then type into the **Search** box on the main



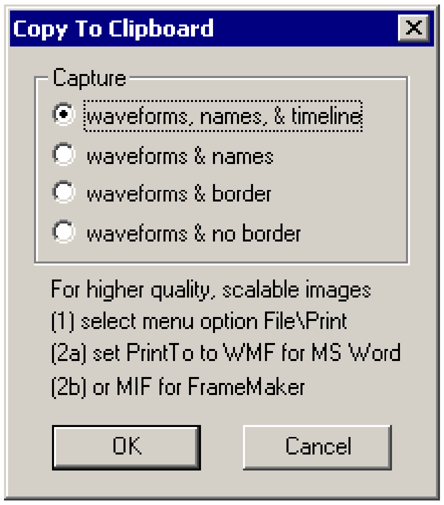
window bar.

- If the Diagram window is selected, then it will search for signal names.
- If a waveform edge is selected in Waveform window, then it will search for a signal value that matches the search string. The search will begin with the signal and edge that is selected and continue from left to right and down the page until a match is found.
- If the Parameter window is selected, then it will search for parameter names.
- If the Report window is selected, then it will search for random text in the selected report tab.

7.3 Images and Word Processors

There are several 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. Also the next section shows how to link an image to a timing diagram file for quick editing using OLE (see Section 7.4 OLE 107). There is also supplemental information on image generation located in **SynaptiCAD > Help> Images.doc,** which compares and contrasts different image types.

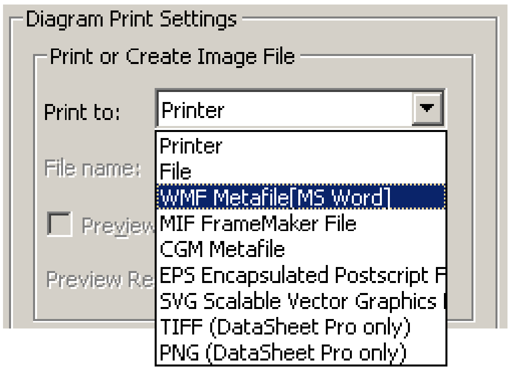
Windows Bitmaps using the Clipboard (Windows Only)



For higher quality, scalable images
(1) select menu option File\Print
(2a) set PrintTo to WMF for MS Word
(2b) or MIF for FrameMaker

- Select the **Edit > Copy Bitmap to Clipboard** menu option to open the dialog.
- Choose what objects to capture. The time range will be the displayed range on the screen. When the dialog is closed the bitmap image will be in the Windows clipboard.
- Inside your graphics or word processing program use the **Edit > Paste** menu command to embed the image into your document.
- Note that this is the lowest quality image generated by the editor, but it is quick and easy to use.

Image Formats using the Print Dialog



- Select the **File > Print Diagram** menu to open the *Print* dialog.
- Choose an Image file format from the **Print to** box (WMF, MIF, CGM, EPS, SVG, TIFF, or PNG).
- Enter a file name into the **File Name** box
- Press the **Ok** button to generate the file.

Format Quality Overview

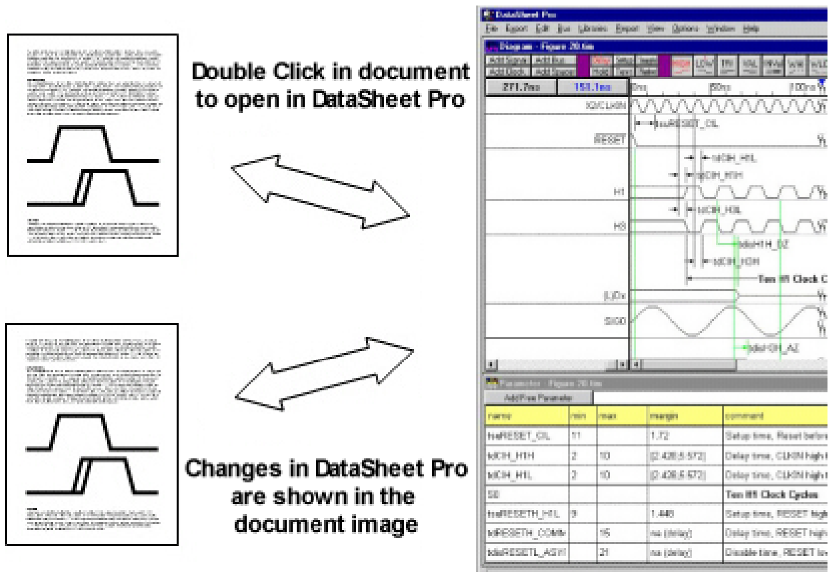
Name	Type	Ease of use	Comments
Copy-to-Clipboard	Window s bitmap	very easy	medium quality images w hich cannot be sized or scaled w ithout losing resolution
WMF metafiles	vector metafile	very easy	excellent quality images w hich can be sized or scaled - recommended for Microsoft Office/Window s Apps
MIF files	vector file	moderately easy	excellent quality images - for use w ith FrameMaker only
SVG files	vector	very easy	w eb-ready, loss-less data compression (free tools available), supports cross platform images
TIFF files	raster	very easy	loss-less data compression, high quality
PNG files (DataSheet Pro)	raster	very easy	w eb-ready, loss-less data compression, high quality
JPEG files (DataSheet Pro**)**	raster	very easy	w eb-ready, lossy data compression
CGM metafiles	Vector metafile	moderately easy	excellent quality images - support for cross-platform images
EPS files	Vector file	most difficult	excellent quality images - recommended for cross-platform images and data books

Special Image Instructions:

- When the **EPS** file is selected, a **preview** check box will be enabled. Embedding a preview into the eps allows the file to be displayed (as a bitmap) within a word processor, but when the document is printed then the vector EPS information is used for printing. 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.
- When the **MIF-FrameMaker** file is selected the **Margins** section of the dialog has different controls that can be used to set the horizontal MIF image size. The **Specify Image Size** enables the width edit box, and the **Width** box sets the width of the MIF image in inches.

7.4 OLE- Object Linking and Embedding

The OLE option allows images to be embeded into word processors and other applications so that the image and the timing diagram file are linked. Double clicking on the image will launch the timing diagram editor for editing. DataSheet Pro includes the OLE option, and it is also available as an add on to WaveFormer Pro, Timing Diagrammer Pro, VeriLogger, BugHunter and TestBencher Pro. This is a Windows Only Feature.


Double Click in document to open in DataSheet Pro
Changes in DataSheet Pro are shown in the document image
DataSheet Pro
Diagram - Figure 20.tim
271.7ns
151.1ns
RESET
H1
H3
Ten ## Clock C
SIGO
Add/Free Parameter

name	min	max	margin	comment
tsuRESET_CIL	11		1.72	Setup time, Reset befor
tdCIH_H1H	2	10	[2.428,5.572]	Delay time, CLKIN high
tdCIH_H1L	2	10	[2.428,5.572]	Delay time, CLKIN high
S0				Ten ## Clock Cycles
tsuRESETH_H1L	9		1.448	Setup time, RESET high
tdRESETH_COMM		15	na (delay)	Delay time, RESET high
tdisRESETL_ASYI		21	na (delay)	Disable time, RESET lo

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.

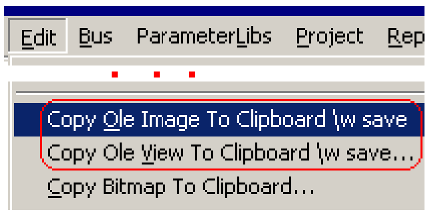
Embedding allows you to save timing diagrams directly into a 3rd party file. If you have a timing diagram you want to show in a report, for instance, you may embed a copy of your timing diagram file directly into your word processing file. Later, you can open the word processing file and edit the embedded image by double-clicking on it, which re-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. Unlike OLE embedding, the timing diagram is stored into a separate file and the 3rd party file contains a link to the timing diagram file. 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.

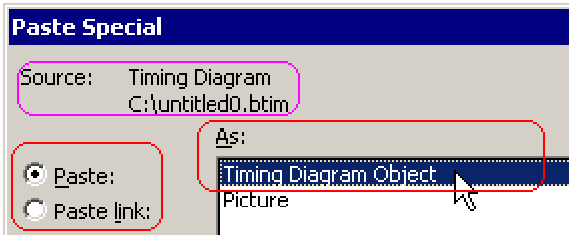
When OLE option is enabled it turns the timing diagram editor into an ***OLE Server***. This means that timing diagrams can be linked to or embedded within other applications. These applications, in turn, are known as ***OLE Clients*** since they make use of the server's features. Many of the more popular 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.

To embed or link a timing diagram into a 3rd party application:

- Inside DataSheet Pro, choose one of the **Edit > Copy OLE** menus to copy the visible timing diagram to the clipboard. Note: If the diagram file has not been previously saved to a valid filename, the *Save As* dialog will appear so that you can assign it a filename.


Edit
Bus
ParameterLibs
Project
Rep
Copy Ole Image To Clipboard \w save
Copy Ole View To Clipboard \w save...
Copy Bitmap To Clipboard...

- In your third-party application (like Word) select **Edit > Paste Special** to open the pasting dialog.
- The following image is from the Microsoft Word's Paste Special dialog but most paste dialogs will have similar controls:
- Check to see that the source is your timing diagram.
- Select the **Paste** radio button (for embedding) or **Paste link** (for linking).
- Select **Timing Diagram Object** from the **As** list.


Paste Special

Source: Timing Diagram C:\untitled0.btim

- Click OK to close the dialog and you should see the embedded timing diagram within your application.

Print Dialog Controls which objects get rendered for OLE objects:

The print dialog controls which objects in the timing diagram get rendered, the print time range, and whether a border box or a time line is included or not. See Section 7.5: Printing[109] for more information on this dialog.

7.5 Printing

Both the timing diagram and the parameter table can be printed directly to a printer or sent to an image file that can be imported into word processors. If you need the parameter table information to be editable when it is imported into a word processor, then it is better to use the export features covered in Section 11.10 Spreadsheets - Import and Export[173].

Print either the Diagram or Parameter table:

- Select either the **File > Print Diagram** or the **File > Print Parameters** menu to open the *Print* dialog. The **Print Window** controls are used to switch between the diagram and parameters options.

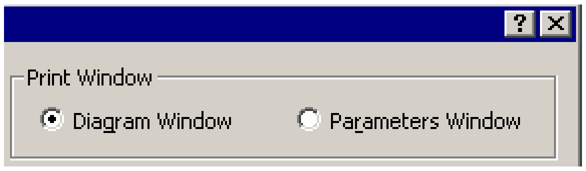
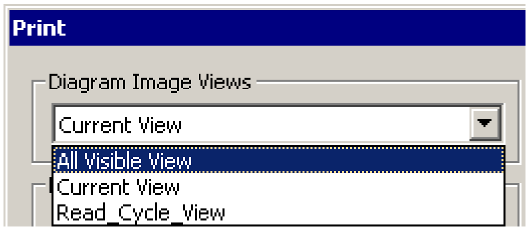


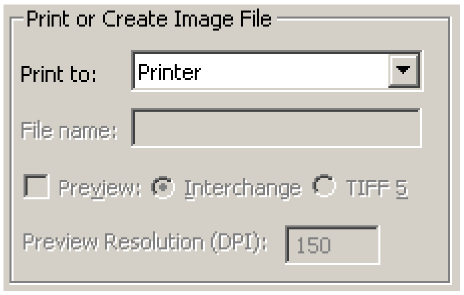
Image Views:

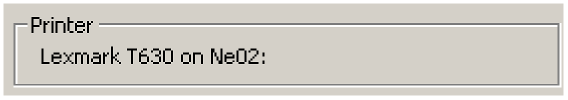
- DataSheet Pro can print different "views" of a diagram (See Section 7.1 Views 103). For the other editors, the **Current View** is all the visible timing diagram or parameters.



Print or Create Image File:

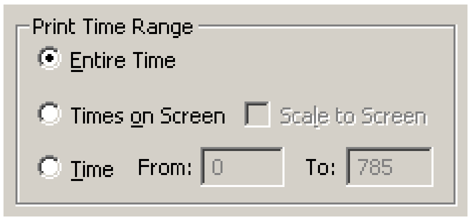
- Use the **Print to** box to pick printer or an image format. **Printer** prints to the default printer, which can be changed using the **Setup** button. **File** prints to a file that is compatible with the default printer. The default printer is displayed in the *Printer* section.
- The image file formats are covered in Section 7.3 Images and Word Processors 106.





Print Time Range:

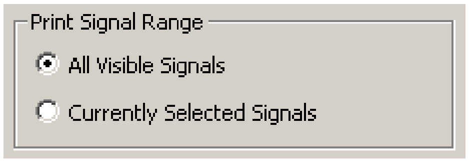
- **Entire Time** prints to the end of the longest drawn signal or the first **End Diagram** marker.
- **Times on Screen** prints the portion of the diagram that currently appears in the *Diagram* window. **Scale to Screen** also scales the text to the image size.



- **Time** prints from the starting time to the ending time.

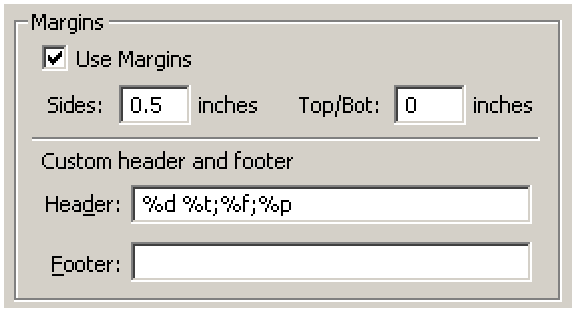
Print Signal Range:

- **All Visible Signals** prints all signals that are not hidden.
- **Currently Selected Signals** prints only signals that are selected or names highlighted in the Label window.



Margins:

- **Use Margins** enables the margin options. If this is unchecked, the only margins will be the minimum area needed by your printer. This option, when combined with an unchecked **Print signal names** check box, is useful if you are trying to tape long diagrams together.
- **Sides** and **Top/bot** indicates how wide the margins should be.

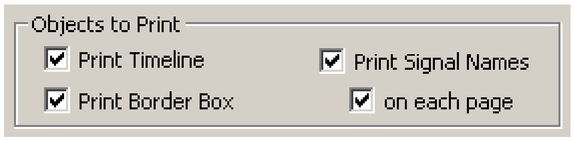


- **Custom header and footer strings** allows text with control codes to be printed in the margins of the paper. Use a semicolon to separate left, center, and right justified text. In the example, the filename will be printed in the center of the header.

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

Objects to Print:

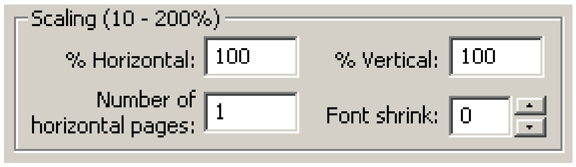
- **Print Timeline** inserts the timeline above the waveforms.
- **Print Border Box** draws a black border around the image.


Objects to Print
☑ Print Timeline ☑ Print Signal Names
☑ Print Border Box ☑ on each page

- **Print Signal Names** prints the names of the signals. The **on each page** control specifies whether to show the names on each page of the printing.

Scaling:

- **% Horizontal** and **% Vertical** indicate the amount to scale the drawing. We recommend you use TrueType fonts when scaling as regular fonts do not scale.

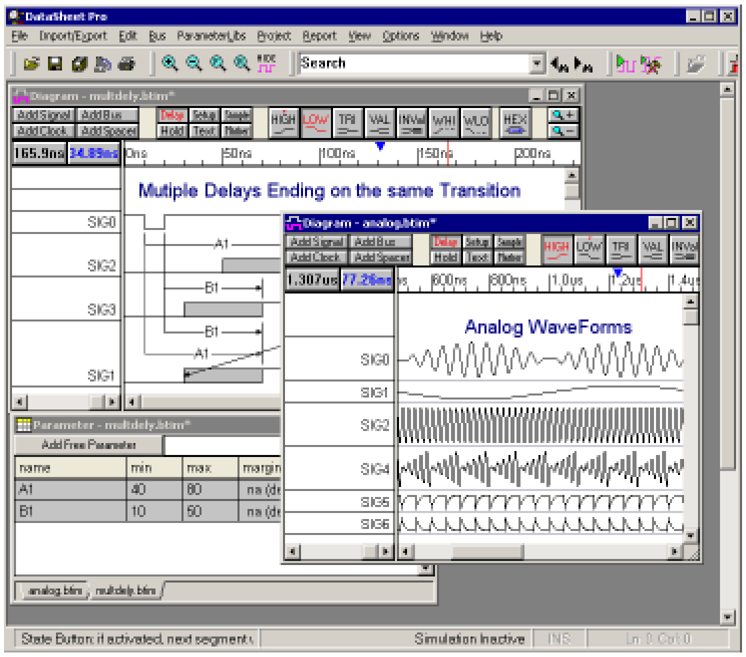

Scaling (10 - 200%)
% Horizontal: 100 % Vertical: 100
Number of horizontal pages: 1 Font shrink: 0

- **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.

7.6 Miscellaneous Diagram Features

Multiple Timing Diagram Windows:

The multiple timing diagram feature allows multiple timing diagrams to be opened in the editor at the same time. This feature is included in DataSheet Pro, and it can be added on to WaveFormer Pro and Timing Diagrammer Pro. The multiple parameter tables are shown as different tabs in the *Parameter* window.


DataSheet Pro
File Import/Export Edit Bus ParameterLibs Project Report View Options Window Help
Search
Diagram - multdely.btim*
Mutiple Delays Ending on the same Transition
SIG0
SIG2
SIG3
SIG1
A1
B1
Diagram - analog.btim*
Analog WaveForms
SIG0
SIG1
SIG2
SIG4
SIG5
SIG6
Parameter - multdely.btim*
Add Free Parameter
analog.btim
multdely.btim
State Button: if activated, next segment
Simulation Inactive

name	min	max	margin
A1	40	80	na (d
B1	10	50	na (d

Drag and Drop File Load

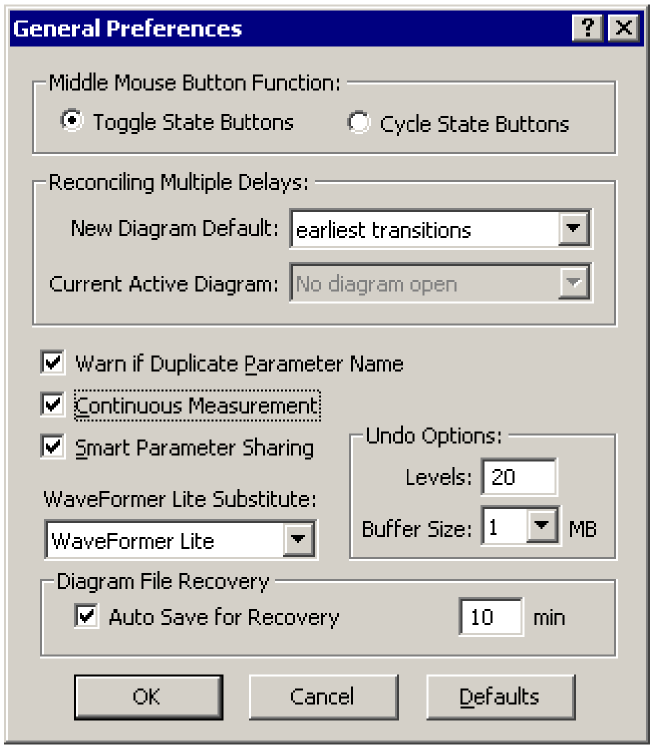
- **Drag-N-Drop File Load:** From Windows Explorer you can drag a *.tim or *.btim file (the program's normal output file) and drop it into the timing window to open the file. This is helpful if you have many timing diagrams and would like to view them in rapid succession.

Read Only Mode:

- The read only mode may be used to ensure that changes to diagram waveforms can not be made when importing and exporting data. To turn on the read only mode, check the the **Options > Read-Only Mode** menu.

Multiple Edit undo and redo:

- To undo or redo an edit use the **Edit > Undo** or **Edit > Redo** menu items.
- The number of undo levels is set in the *General Preferences* dialog. This is setup to have a high level of undos for small timing diagrams. If your diagrams are very large, you cave memory by setting this to a lower number. To open the dialog choose the **Options > General Preferences** dialog.


General Preferences

Middle Mouse Button Function: [✓ Toggle State Buttons] Cycle State Buttons

Reconciling Multiple Delays:

New Diagram Default: earliest transitions

Current Active Diagram: No diagram open

[✓] Warn if Duplicate Parameter Name

[✓] Continuous Measurement

[✓] Smart Parameter Sharing

WaveFormer Lite Substitute: WaveFormer Lite

Undo Options:

Levels: 20

Buffer Size: 1 MB

Diagram File Recovery

[✓] Auto Save for Recovery 10 min

Auto Save and Recovery of Data

The *General Preferences* dialog (see above picture) also controls the Auto Save for Recovery feature. Timing diagrams will be temporarily saved at the indicated interval. If the program is prematurely terminated, then the next time it is launched it will reload the temporary file so that you can choose to use this data or revert to the original file. This is product specific feature, so you must launch the same program as the program that was running during the crash to recover the file. For example, if you were running TestBencher before your operating system crashes, then you must run TestBencher (not WaveFormer or other products) to recover the data.

- Choose **Options > General Preferences** to open the dialog.
- Check the **Auto Save for Recovery** box to enable the feature and type in time interval in which to do saves.

Project Window for DataSheet Pro:

DataSheet Pro has a project window that can hold timing diagram files which makes it quick to cycle through a lot of timing diagrams.

- Choose **Project > New Project** to open a new project window.
- Right click in the project window and choose **Add New Timing Diagram** to add timing diagrams.
- To view a timing diagram just double click on the diagram name.
- Choose **Project > Save Project** to save your project file, and **Project > Load Project** to open it again.

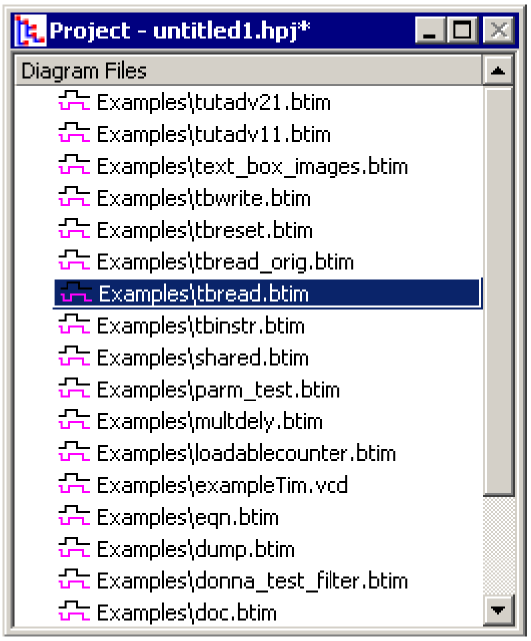
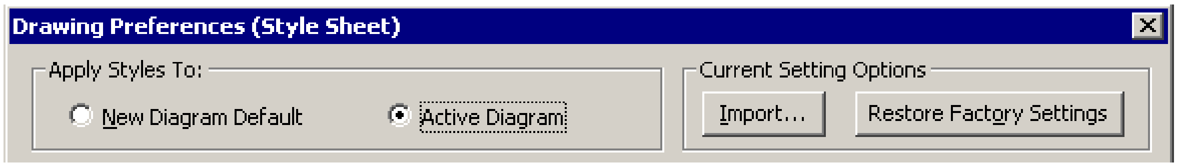

Project - untitled1.hpj*

Diagram Files
- Examples\tutadv21.btim
- Examples\tutadv11.btim
- Examples\text_box_images.btim
- Examples\tbwrite.btim
- Examples\tbreset.btim
- Examples\tbread_orig.btim
- Examples\tbread.btim
- Examples\tbinstr.btim
- Examples\shared.btim
- Examples\parm_test.btim
- Examples\multdely.btim
- Examples\loadablecounter.btim
- Examples\exampleTim.vcd
- Examples\eqn.btim
- Examples\dump.btim
- Examples\donna_test_filter.btim
- Examples\doc.btim

Style Sheets and Importing styles:

The *Drawing Preferences* dialog controls how objects appear in the *Diagram* window, and essentially creates a style for a timing diagram. This style can be imported from a one timing diagram into a different timing diagram.

- To open this dialog, select the **Options > Drawing Preferences** menu option.


Drawing Preferences (Style Sheet)

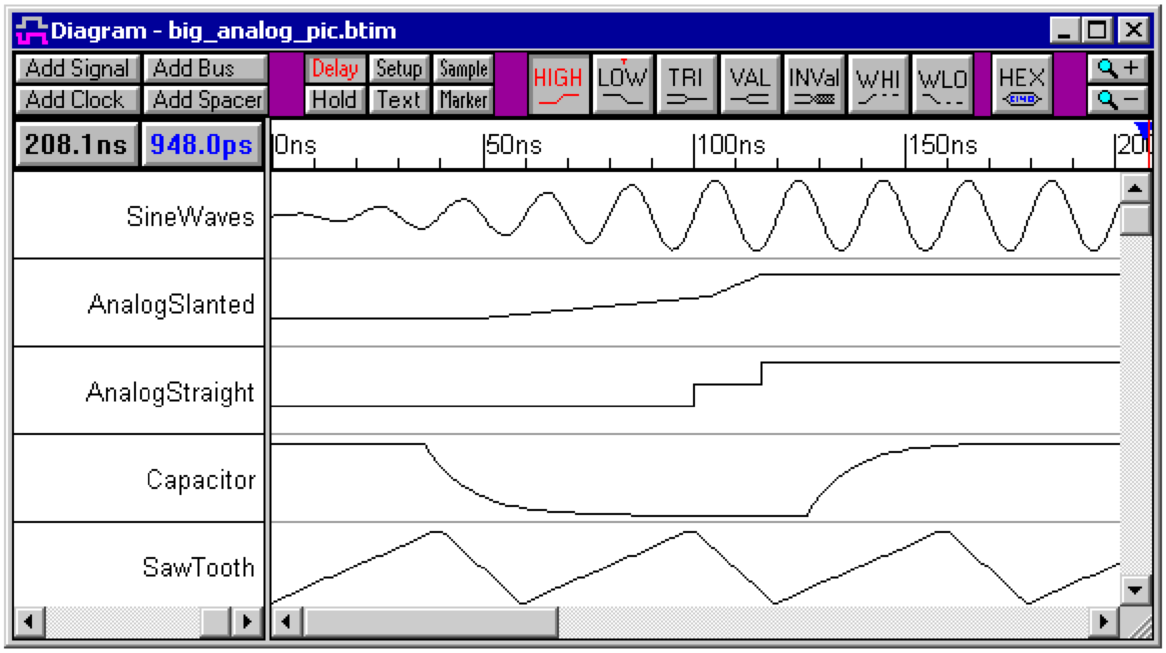
Apply Styles To: New Diagram Default [✓ Active Diagram]

Current Setting Options

- Press the **Import** button to open a file dialog that lets you load the style information from another timing diagram.
- Choose **New Diagram Default Style** to modify the style settings for all new timing diagrams.
- Or choose **Active Diagram Style** to modify the style settings of the active timing diagram.

Chapter 8: Analog Display and SPICE

The analog features allow analog signals to be imported, exported, created, and manipulated. Digital signals can be converted to analog signals and vice versa (similar to the way analog-to-digital converters operate). There is also a tutorial called Creating Analog Waveforms that will help you work through some of the features.

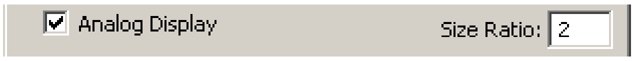


8.1 Analog Waveform Display

To display a signal as an analog waveform, the editor will use the signal's **logic high/low voltage** to determine the high and low value for the range. Analog signals can be displayed from either a real-valued signal or a multi-bit digital signals (in which case the Analog Display acts like the signal's data being passed to a D/A Converter). If the signal has a radix that is not **real** (it's a multi-bit digital signal), then the **MSB/LSB** will be used to determine the resolution of the quantization of the digital value and the high/low voltages in the *Analog Properties* dialog determine the range of the analog output of the D/A converter.

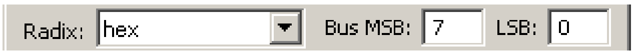
Display Analog Waveforms:

- Double-click on the signal name to open the *Signal Properties* dialog, and check the **Analog Display** box.
- To increase the vertical height of the signal, set the **Size Ratio** box to be greater than 1.



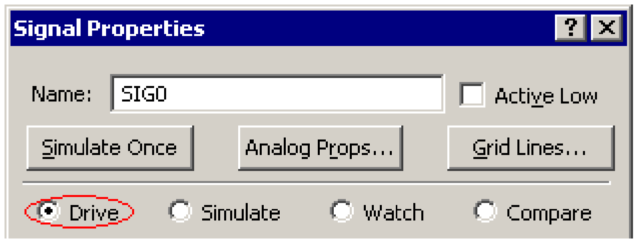
[✓] Analog Display Size Ratio: 2

- If radix is NOT **real**, then make sure to set the **MSB** and **LSB** to the proper range for the signal.

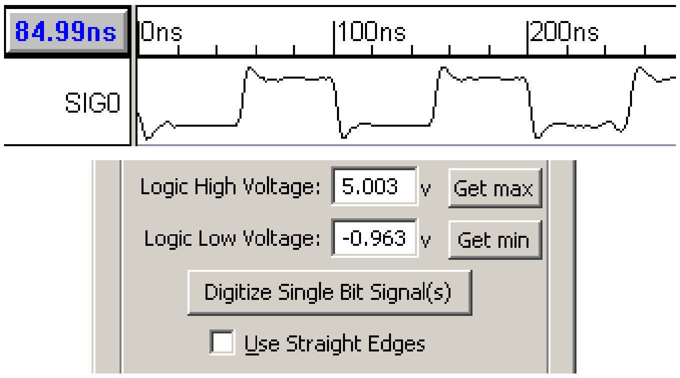


Radix: hex Bus MSB: 7 LSB: 0

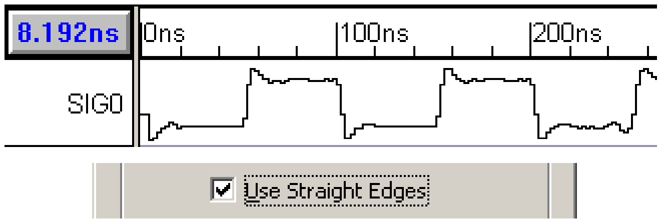
- Press the **Analog Props** button to open the *Analog Properties* dialog



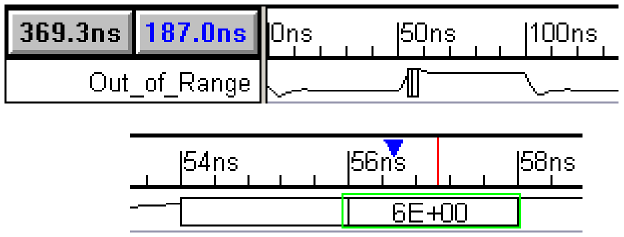
- Set the **Logic High Voltage** and **Logic Low Voltage** boxes to specify the minimum and maximum values of an analog signal.
- If the radix of the signal is real, the **Get Min** and **Get Max** buttons will set the appropriate high/low voltage according to the values found on the signal.


84.99ns
0ns
100ns
200ns
SIG0
Logic High Voltage: 5.003 v
Get max
Logic Low Voltage: -0.963 v
Get min
Digitize Single Bit Signal(s)
Use Straight Edges

- Normally a slanted line is drawn between the points in an Analog signal. However, if you check the **Use Straight Edges** setting a vertical line will be used.



- If a value on the signal is above or below the voltage range, then it will be displayed as a box. If you, zoom in the box will display the value that is out of range.



Converting between Real and Multi-Bit Analog Signals

Sometimes it's useful to convert between real-valued and multi-bit digital signals. Some examples of real-valued signals include data imported from SPICE simulations and signals created using the built-in analog label equations (discussed in the next sections). One example where it can be useful to convert these signals to multi-bit digital signals is if you want to export them to Verilog digital stimulus.

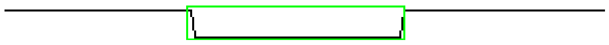
Converting between real and multi-bit signals is done by changing the **radix** and the MSB and LSB as shown above. Before converting, it is a good idea to look at the *Analog Properties* dialog to make sure the voltage **min**, **max** and **range** values are what they need to be. These values will control the conversion between the signals.

8.2 Waveform Equation Blocks for editable Analog waveforms

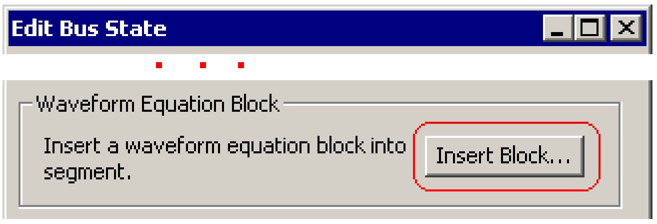
Sine, Capacitor, Ramp and Exponential waveforms can be inserted into a waveform segment using a Waveform Equation Block. Unlike the State Label equations of the next section which append analog waveform segments onto the end of a signal, waveform equation blocks can edited after they are created. For this reason, it's often better to use a waveform equation block rather than a State Label equation.

Overview of steps to add a Waveform Equation Block to a Signal

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



- Press the **Insert Block** button to open the open the *Waveform Equation Block Properties* dialog.

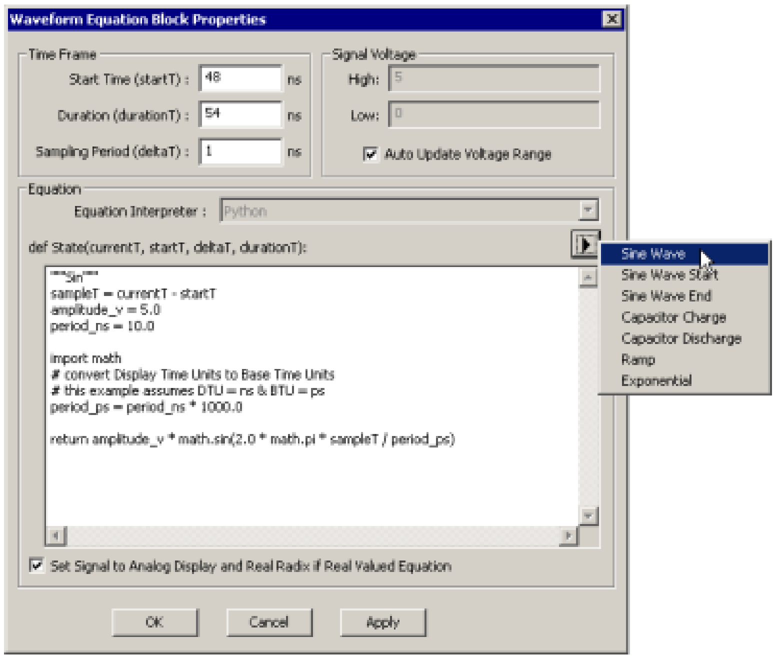


Edit Bus State

Waveform Equation Block

Insert a waveform equation block into segment.

- Use the fly-out to pick one of the preprogrammed analog waveforms or write your own Python function.
- Notice that you can edit the **period** and **amplitude** directly in the Python code.
- The fields of this dialog are described below in the ***Programming Waveform Equation Blocks*** sub-section.
- Press OK to exit this dialog and then close the Edit Bus State dialog.



Waveform Equation Block Properties

Time Frame

Start Time (startT) : 48 ns

Duration (durationT) : 54 ns

Sampling Period (deltaT) : 1 ns

Signal Voltage

High: 5

Low: 0

✓ Auto Update Voltage Range

Equation

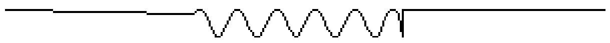
Equation Interpreter : Python

def State(currentT, startT, deltaT, durationT):

```
"""Sin"""
sampleT = currentT - startT
amplitude_v = 5.0
period_ns = 10.0

import math
# convert Display Time Units to Base Time Units
# this example assumes DTU = ns & BTU = ps
period_ps = period_ns * 1000.0

return amplitude_v * math.sin(2.0 * math.pi * sampleT / period_ps)
```

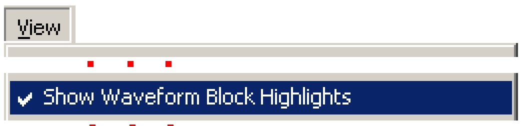
✓ Set Signal to Analog Display and Real Radix if Real Valued Equation

- A sine wave inserted between two high segments will look like this.

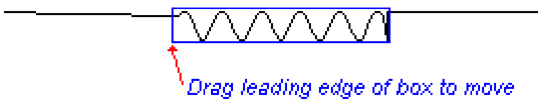


Editing a Waveform Equation Block

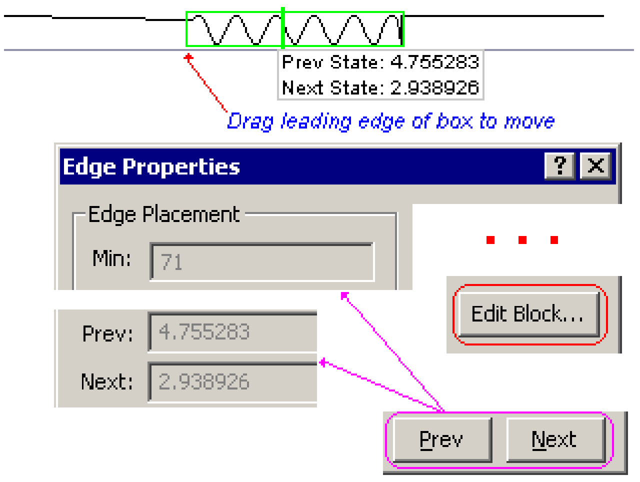
- Checking the **View > Show Waveform Block Highlights** menu will draw blue boxes around all of the Waveform Equation Blocks.



- Double clicking inside the blue box will open the *Waveform Equation Block Properties* dialog so you can edit the equation.

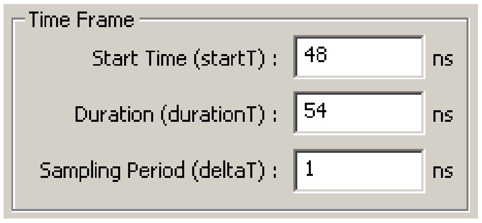


- If the blue boxes are not displayed, then double clicking will open the *Edge Properties* dialog. From here you can press the **Edit Block** button to open the *Waveform Equation Block Properties* dialog.
- Equation Blocks cannot be edited from the *Edge Properties* dialog, but you can use the Previous and Next buttons to investigate events generated by the waveform equation block.
- The first edge of a blue or green box can be dragged and dropped to move the entire block.



Programming Waveform Equation Blocks

- **Start Time** sets where the block begins. The first event generated by the block will be at StartTime + Sampling Period.
- **Duration** is the length of the block. The duration will always be truncated to be an integral number of sampling periods, so the last event in the block will be at Start Time + Duration (e.g. the end of the block).
- **Sampling Period** is the time delta between events in the block. The smaller the sampling period, the more points that are generated for the block.



Start Time (startT) : 48 ns
Duration (durationT) : 54 ns
Sampling Period (deltaT) : 1 ns

- **State** is the name of a Python function call that is evaluated for each event in the block to generate the extended state for that event. You define the body of this function to control what the block looks like. The parameters can be referenced in the function's body.

```
def State(currentT, startT, deltaT, durationT):
```

currentT is the current time for the event (e.g currentT will be startT + deltaT for the first event and startT + durationT for the last event in the block).

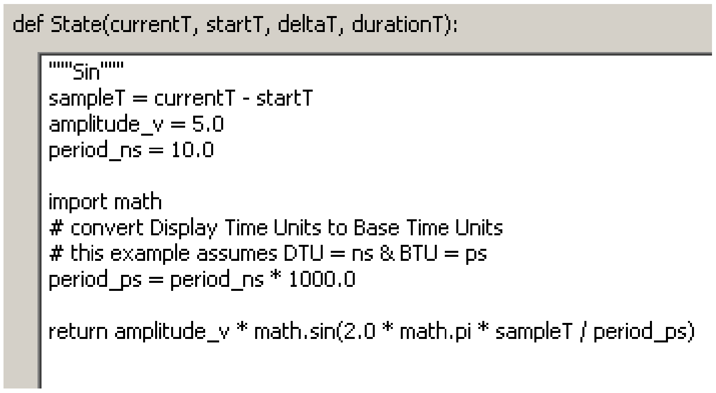
startT is the start time for the block.

deltaT is the sampling period for the block.

durationT is the duration of the block.

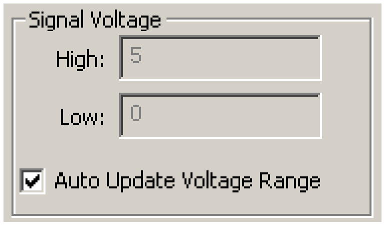
- You can type in any Python code or edit the code of the default analog functions.
- **import math** reads in the standard Python library code.
- Most of the default functions have the statement ***sampleT = currentT-startT***, which means that the analog block will have the same waveform shape regardless of the starting time of the block. To use absolute times instead of relative times for state calculations, just change this line to read:

  ```
  sampleT = currentT
  ```

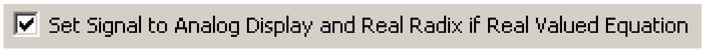


```
def State(currentT, startT, deltaT, durationT):

"""Sin"""
sampleT = currentT - startT
amplitude_v = 5.0
period_ns = 10.0

import math
# convert Display Time Units to Base Time Units
# this example assumes DTU = ns & BTU = ps
period_ps = period_ns * 1000.0

return amplitude_v * math.sin(2.0 * math.pi * sampleT / period_ps)
```

- The values in the **Signal Voltage** section are taken from the *Analog Properties* of the signal as described in Section 8.1: Analog Waveform Display 115.
- **Auto Update Voltage Range** scans the entire signal and sets the maximum and minimum voltage values. This is equivalent to pressing the **Get Min** and **Get Max** buttons in the Analog Properties dialog.



Signal Voltage
High: 5
Low: 0
☑ Auto Update Voltage Range



☑ Set Signal to Analog Display and Real Radix if Real Valued Equation

- Checking this control automatically sets the **Analog Display** box of the parent signal and sets the signal radix to **Real** so you do not have to manually set it. This only happens if the State function is written to generate 'real' values (State functions can also be used to generate digital patterns, in which case this check box has no effect).

Advanced Waveform Equation Programming

Waveform Equation Blocks can access additional information about the timing diagram through our **BtimPython** module. While a Waveform Equation Block's code is executing, it can access the information about the waveform block through the global object **currentWaveformBlock**. From this object, you can navigate to objects representing the signal containing the block, the btim document, and other signals and event data in the document. This allows equations to generates state values based on either the event values of other signals or previous event values in the same signal.

There are several examples of equations which use the **BtimPython** API in the example file called **waveform_block_equations.btim**. The signals named **from_text_file**, **product**, and **ramp_max_to_min** demonstrate using the API to access the document's filename, get events from other signals, and access properties of the containing signal.

The full BtimPython API is documented in the file **btim.html** located in the **SynaptiCAD/help** directory. The BtimPython module is designed for use in stand-alone python programs as well as Waveform Equation Blocks. While all of the API functions can be called from a waveform equation block, many of the functions available in the module (such as creating new timing diagrams) will not be useful in this context.

Basing an Equation on the Preceding State Before the Waveform Block

It is often useful to define a waveform block equation so that it's waveform is a function of the state of the signal just before the waveform block begins. Two Python functions are available to get this preceding value, depending on whether the waveform block is generating analog (real values) or digital values:

- **GetPrecedingRealValue()** returns the state of the preceding event as a real value suitable for analog equations.
- **GetPrecedingState()** returns the state of the preceding event as a string value (generally this will be a binary string) suitable for digital equations.

The signals called **uses_preceding_event** and **uses_preceding_digital** in the **waveform_block_equations.btim** file contain waveform block equations that demonstrate usage of these two functions.

Formatting the Equation's Return Value for Digital Signals

When using Waveform Equation Blocks to generate analog equations, the python code fragment should return a python floating point value. Waveform Equation Blocks can also be used to generate digital signals. For digital signals, your Python code should return values as a strings of binary digits so that the signal's radix can post-process the value. For example, if you wish to return the hexadecimal value 'h71, you would want your python code to say: "return '1110001'". When the values are displayed to the user, they will be converted using the radix set in the Signal Properties dialog. For example, if the signal's radix is set to "hex" and the MSB:LSB is set to 15:0, that string will be formatted as 0071.

Two different ways of converting your digital values to binary strings are demonstrated in the signals **random** and **repeating_sequence** in **waveform_block_equations.btim**. In the signal repeating_sequence, a decimal value is converted to a binary string using this python command:

```
  valueAsBinary = '{0:b}'.format(value)
```

8.3 Sine, Capacitor, Ramp & Exponential waveforms

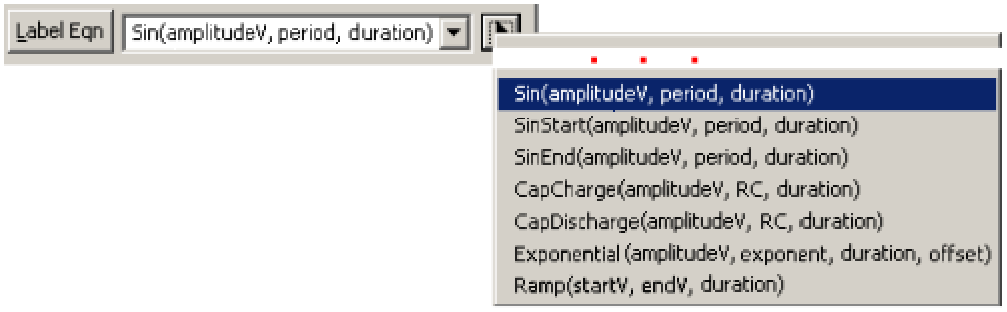
Sine, Capacitor, Ramp, and Exponential waveforms can be automatically generated using State label equations. These are special state label equations that generate both the waveform and the state values at the same time. See Section 1.3 Generating Waveforms and States with Equations 18 for general information on label equations.

This is an older method of generating analog waveform data and these equations generate "one-time" results (the generated events react thereafter the same as if you had manually drawn these events). Unlike the Waveform Equation Blocks described in the previous section (Section 8.2: Waveform Equation Blocks for editable Analog waveforms 116), the original equation cannot be changed to change the waveform, only the individual events can be edited.

For all the functions shown in this section, there is also an optional last parameter that can specify the number of **points** to use when drawing the waveform. For example, the first equation could also be written as **SinStart(5,20,100,40)** to indicate that 40 points should be used to draw the waveform. If no points are specified the tool will decide how many points to use. The sampling frequency can also be forced to be a particular value for all new signals when the **Enforce as Sampling Period** check box is checked in the the *Edit Text and Edge Grids* dialog.

Generate the Analog State Label equations:

- Double-click on a signal's name to open the *Signal Properties* dialog.
- Use the QuickFill button (black triangle) to quickly add one of Analog State Label equations to the Label Equation edit box.

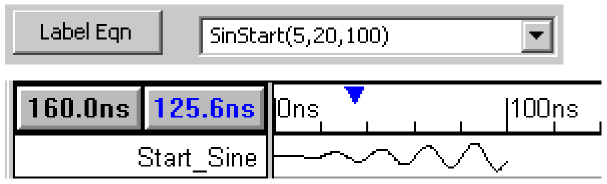

Label Eqn
Sin(amplitudeV, period, duration)
Sin(amplitudeV, period, duration)
SinStart(amplitudeV, period, duration)
SinEnd(amplitudeV, period, duration)
CapCharge(amplitudeV, RC, duration)
CapDischarge(amplitudeV, RC, duration)
Exponential (amplitudeV, exponent, duration, offset)
Ramp(startV, endV, duration)

- Replace the dummy parameters in the equation with values that will generate the waveform.
- Press the **Label Eqn** button to generate the waveform.

Sine Wave Equations

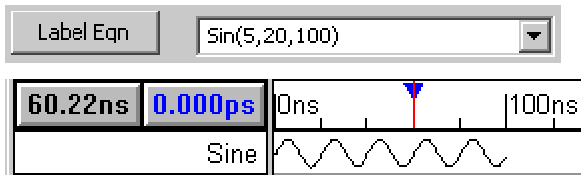
- A **SinStart** equation will generate a waveform with initial state at zero and that slowly increases in amplitude over the duration specified.

```
SinStart([amplitudeV], [period], [duration])
```



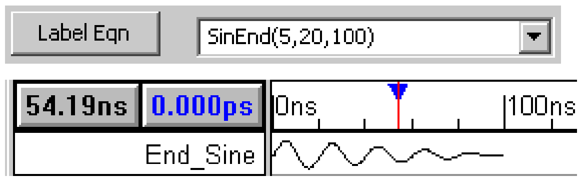
- A **Sin** equation will generate a regular sine wave using the amplitude, period, and duration specified.

```
Sin([amplitudeV], [period], [duration])
```



- A **SinEnd** equation will generate a waveform that starts with the full amplitude specified and slowly decreases to amplitude zero over the specified duration. The rate at which it decreases depends on the amplitude and duration specified.

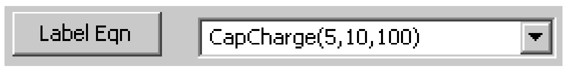
```
SinEnd([amplitudeV], [period], [duration])
```



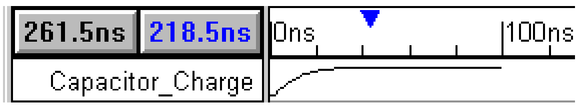
Capacitor Equations

- The **CapCharge** function generates a waveform that represents a charging capacitor. The initial state will have the lowest value specified by the amplitude. The value will then slowly grow at a rate controlled by

```
CapCharge([amplitude], [RC], [duration])
```

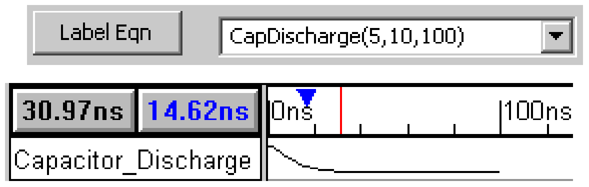


the value of RC (the circuit's RC time constant). The generated waveform will be for the specified duration, regardless of whether or not the 'capacitor' has reached its full charge by the end of the specified duration.



- The **CapDischarge** function performs the inverse of the CapCharge function. A waveform will be generated that represents a capacitor discharging. The initial state will be the peak value specified by the amplitude. The rate of discharge is controlled by the value of RC. The generated waveform will be controlled by the duration specified. The end of the generated waveform will be at the end of the duration regardless of whether or not the 'capacitor' is fully discharged.

```
CapDischarge([amplitude], [RC], [duration])
```

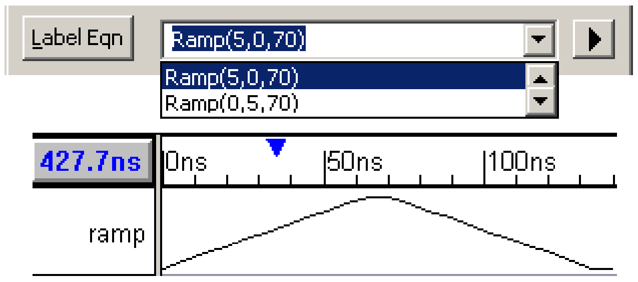


CapDischarge(5,10,100)

Ramp Waveforms

- The **Ramp** equation generates a ramp. In the picture the Ramp was applied once going up and then again going down.

```
Ramp([StartV], [EndV], [Duration])
```



Ramp(5,0,70)

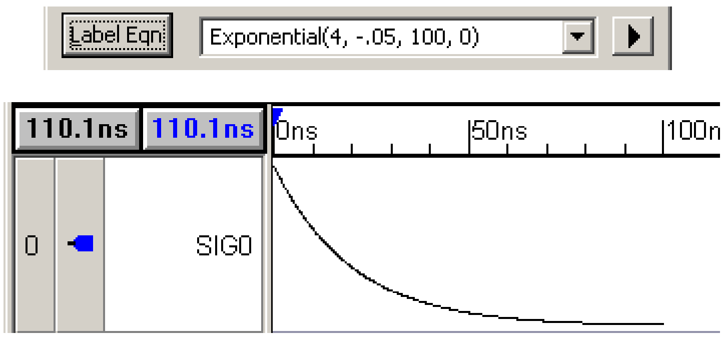
Ramp(5,0,70)

Ramp(0,5,70)

Exponential Waveforms

- The **Exponential** equation generates an exponential waveform.

```
Exponential(amplitudeV, exponent, duration, offset)
```



Exponential(4, -.05, 100, 0)

Setting the Sampling Frequency of an Analog Label Equation

You can set the sampling frequency when you create an analog signal using the optional numberOfPoints parameter. If not specified, numberOfPoints defaults to a value that tends to yield a relatively smooth curve. But when you have a specific sampling frequency requirement for your analog signal, you will need to override the default. Below is an example using the Sin() function.

To create a 3V sine wave with frequency of 5MHz, a sampling frequency of 100MHz and 400 ns duration, the following calculations must be performed:

```
period = 1/frequency = 1/5MHz = 200 ns
sampling period = 1/sampling frequency = 1/100MHz = 10 ns
numberOfPoints = duration/sampling period = 400 ns / 10 ns = 40
```

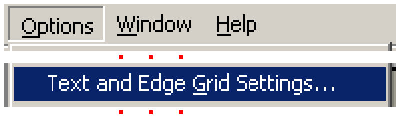
The final equation to create this signal would be:

```
//Sin(amplitudeV, period, duration,numberOfPoints)
Sin(3, 200, 400,40)
```

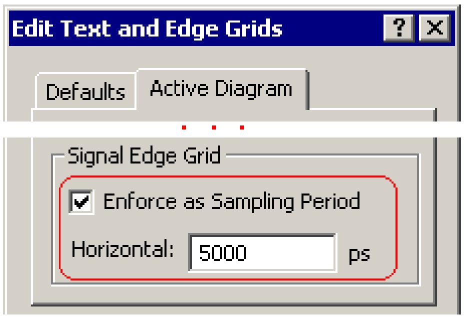
Enforcing the Sampling Frequency of Analog Label Equations:

Normally, the analog label equations pick a sampling frequency that will result in a smooth curve. This often results in a waveform with a lot of points very close together. There is a way to force the sampling period to be a specific time interval.

- Choose the **Options > Text and Edge Grid Settings** menu to open the *Edit Text and Edge Grids* dialog.

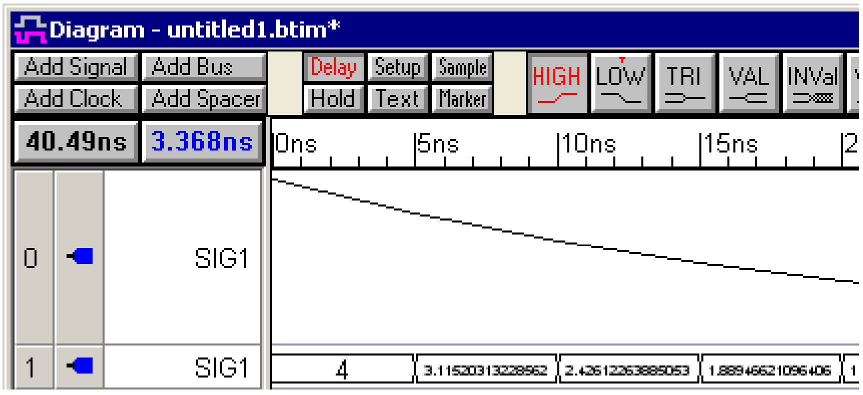


- Check the **Enforce as Sampling Period** check box to take control of the sample period away from the equation generator.
- Type the sampling period into the **Horizontal** box. This will force analog equations to generate evenly spaced points on this grid. Notice that this will also force new digital signal waveforms to be drawn on this grid, so you may wish to change the settings after generating the analog waveforms.



☑ Enforce as Sampling Period
Horizontal: 5000 ps

- In this example, the exponential for SIG1 was created while the sampling period was forced to 5000 ps. Note that the "digitally displayed" version of SIG1 (below the "analog displayed" version) has a state value every 5000 ps.

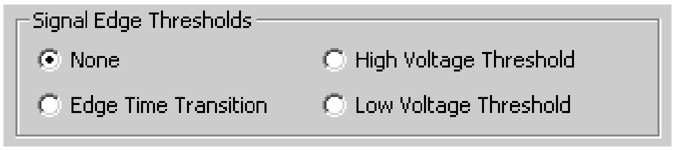


8.4 Display Edge Switching Thresholds

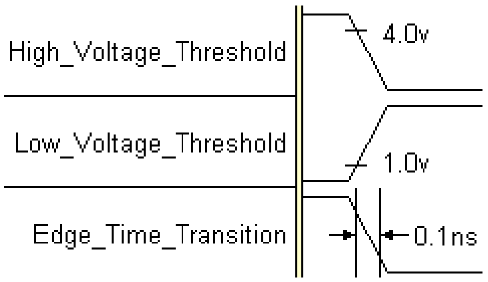
Text Objects that are attached to edges can be made to show the high and low voltage thresholds, and the edge time transition points. The signal's analog properties determine the placement of the marking lines and the values of the properties.

To display the switching thresholds or edge transition times:

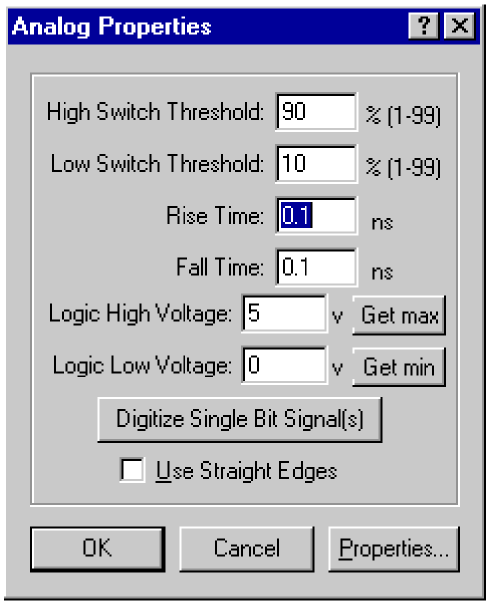
- Add a text object that is attached to the edge (see Section 6.1 Text[91]), then double click on the text to open the *Edit Text* dialog. Select the type of display that you want from the **Signal Edge Thresholds** section. If this section is greyed out, then it means that the text object is not **attached to an edge.**


Signal Edge Thresholds
[● None] ○ High Voltage Threshold
○ Edge Time Transition ○ Low Voltage Threshold

- Checking the **High Voltage Threshold** box creates a tick line at the high voltage level indicated in the *Analog Properties* dialog and displays the voltage. It is similar for the other two settings.



- To adjust the analog settings, double click on the signal name and press the **Analog Prop** button to open the *Analog Properties* dialog.
- The placement of the marking lines is set using the **High Switch Threshold** and **Low Switch Threshold** settings. By default these are set at 90% and 10%.
- The value of the display is set using the **Logic High Voltage** and **Logic Low Voltage** settings. For example a 90% of 5v is 4.5v and that will be the default value of a text object displaying a high voltage threshold.


Analog Properties

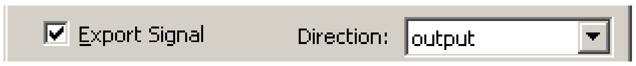
High Switch Threshold: 90 % (1-99)
Low Switch Threshold: 10 % (1-99)
Rise Time: 0.1 ns
Fall Time: 0.1 ns
Logic High Voltage: 5 v
Logic Low Voltage: 0 v
☐ Use Straight Edges

8.5 Exporting SPICE Stimulus

WaveFormer Pro can export waveforms as SPICE voltage sources to provide stimulus for a SPICE netlist. Signals with ***hexadecimal*** and ***binary*** radixes are converted to real-valued numbers before being exported (the voltage range of the output is set in the *Analog Properties* sub-dialog of the *Signal Properties* dialog). 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.

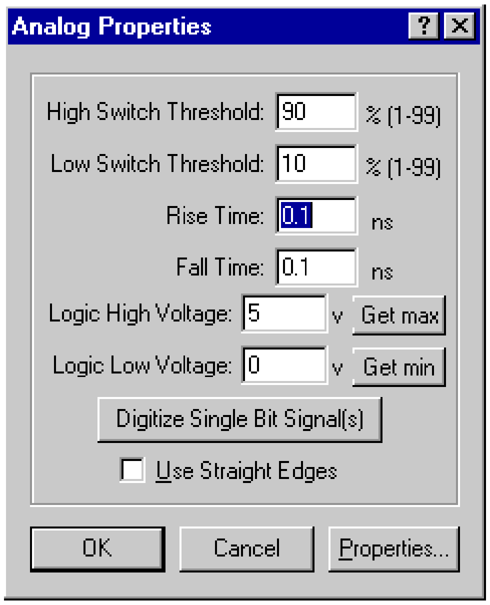
Set the Export and Analog Properties for each signal:

- Select one or more signal names, right-click on a name and choose the **Edit Selected Signals** menu to open the *Signal Properties* dialog. Check the **Export Signals** box.



☑ Export Signal Direction: output

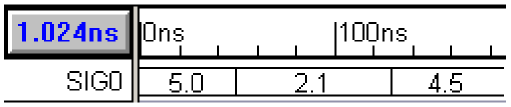
- If the all the signals have a radix of **real** then you are ready to export.
- If the signal radixes are NOT **real**, then press the **Analog Props** button to open the *Analog Properties* dialog.
- Set the analog voltage range for the signals using the **Logic High Voltage** and **Logic Low Voltage** boxes.
- The **High Switch Threshold** and **Low Switch Threshold** specifies the high and low detection limit. These percentages determine the point at which the value will be read as the voltage high or voltage low value.
- The **Rise Time** and **Fall Time** boxes specify the amount of time it takes for the signal to go from the **High Switch Threshold** to the **Low Switch Threshold.**



Analog Properties

High Switch Threshold: 90 % (1-99)
Low Switch Threshold: 10 % (1-99)
Rise Time: 0.1 ns
Fall Time: 0.1 ns
Logic High Voltage: 5 v
Logic Low Voltage: 0 v
☐ Use Straight Edges

- The **Use Straight Edges** check box controls whether the SPICE stimulus exports as piecewise linear segments or as stepped voltages. Piecewise linear is more appropriate when modeling pure analog signals, but stepped voltages are more appropriate for modeling the output of a Digital to Analog Converter circuit (DAC).



Generated Spice code (Not using straight edges)

```
VSIG0 SIG0 0 PWL(
 + 0ns 5.0
 + 49.0725ns 2.1
 + 129.06ns 4.5
 + )
```

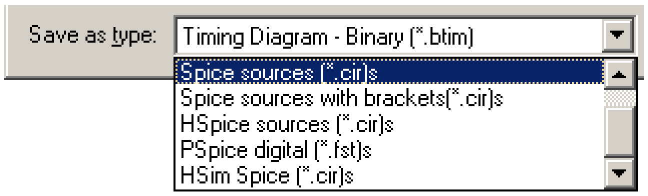
Generated Spice code (Using straight edges)

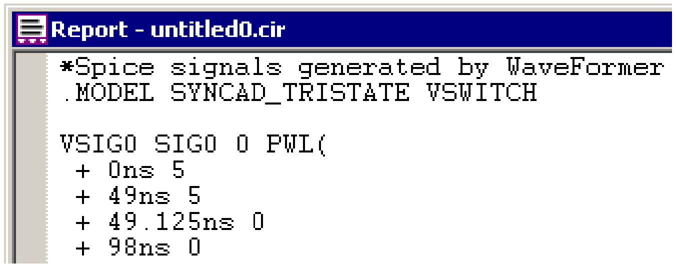
```
VSIG0 SIG0 0 PWL(
 + 0ns 5.0
 + 49ns 5.0
 + 49.0725ns 2.1
 + 129ns 2.1
 + 129.06ns 4.5
 + )
```

Copyright © 2011, SynaptiCAD Sales, Inc.

Export the Signals:

- Select the **Import/Export > Export Timing Diagram As** menu to open the *Save As* dialog.
- Select a SPICE file type to export from the **Save as type** drop-down box, enter a file name, and press the **Save** button to export the file.
- Notice that you can view the generated file in the *Report* window tab. If you cannot see the *Report* window, then choose the **Window > Report** menu option to bring it to the front





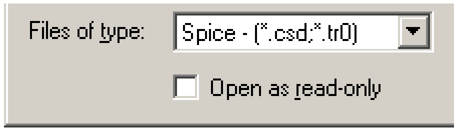
```
*Spice signals generated by WaveFormer
.MODEL SYNCAD_TRISTATE VSWITCH

VSIG0 SIG0 0 PWL(
 + 0ns 5
 + 49ns 5
 + 49.125ns 0
 + 98ns 0
```

8.6 Importing SPICE waveforms

The time-based waveform results of SPICE transient simulation runs (*.csd) can be imported into WaveFormer. It also supports import of binary TR0 files generated by HSPICE and OmegaSim. When signals are imported from a SPICE simulation, all of the values are read in as ***real*** values. The 'Analog Display' checkbox is checked by default for imported signals so that they display as analog waveforms. The logic high and low voltages in the *Analog Properties* dialog box are calculated automatically for waveforms loaded from .csd files (these initial voltage settings can be overridden if desired).

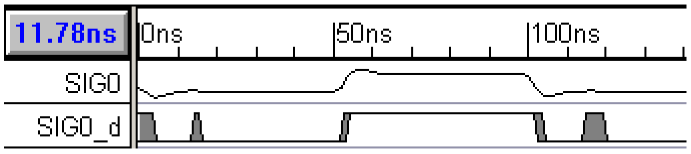
To Import SPICE Waveforms:

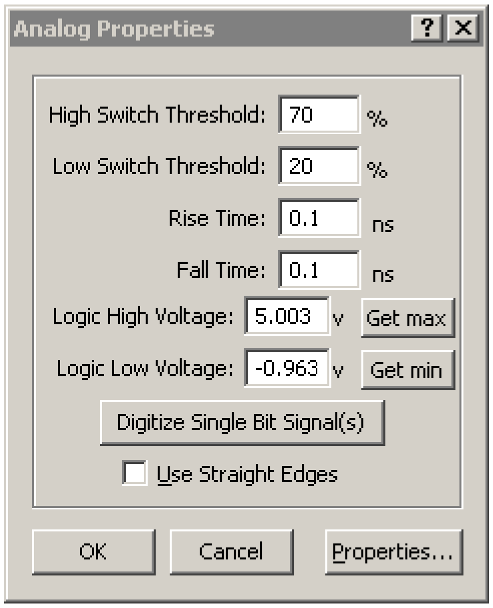
- Select the **Import/Export > Import Timing Diagram From** menu to open the *Open* dialog.
- Select the **SPICE (*.csd, *.tr0)** option from the **Files of Type** box, enter the file name, and press **Open** to import the file.



Converting an Analog Waveform into a Digital Waveform:

- In the Signal Properties dialog press the **Analog Props** button to open the *Analog Properties* dialgo.
- Pressing the **Digitize Single Bit Signals** button creates a new digital single by digitizing the analog signal. Values above the logic high threshold are set to 1, values below the logic low threshold are set to 0, and values between display as uncertainty regions.



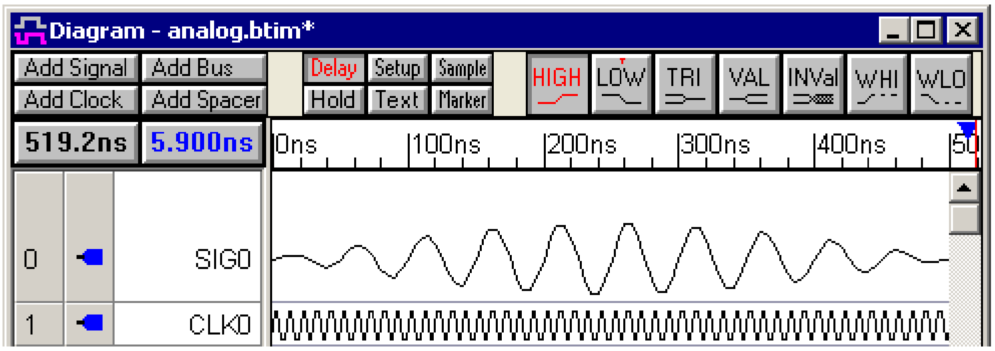


High Switch Threshold: 70 %
Low Switch Threshold: 20 %
Rise Time: 0.1 ns
Fall Time: 0.1 ns
Logic High Voltage: 5.003 v
Logic Low Voltage: -0.963 v
☐ Use Straight Edges

8.7 Changing the Sampling Frequency of Waveforms

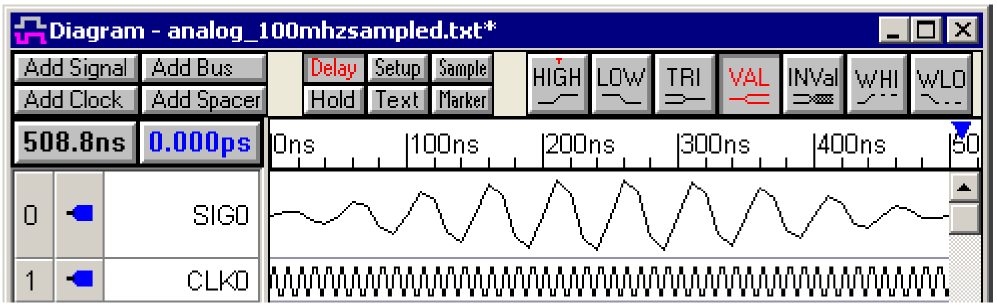
After importing analog signals it is sometimes necessary to export them using a different sampling frequency. To resample signals in a diagram, first add a clock signal with a frequency that equals the new sampling frequency. Then export the waveforms to the spreadsheet format and then re-load the newly sampled signals.

- Add a clock signal to the diagram of the desired sampling frequency. When you export, the other waveforms will be sampled on the positive edge of this sampling clock.
- Select **Import/Export>Set Export Clocking Signal** menu to open a dialog, and set to the newly created clock signal.



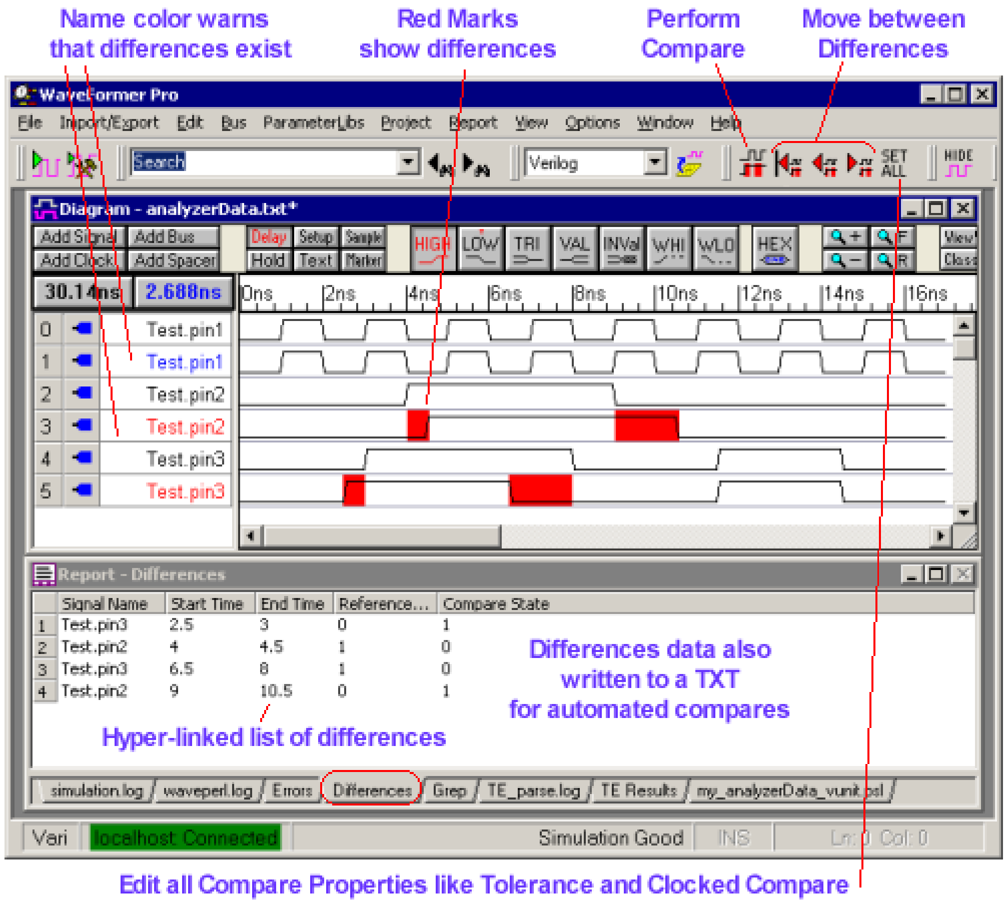
- Export the diagram as a "**Test Vector SpreadSheet Clocked**" (the Clocked keyword here is very important as there are other test vector export formats). This new file will have have points at the sampling frequency of the clock.

- Import the file exported in the previous step as a "**TestVector Spreadsheet**" file to see the signals with the newly sampled waveforms.



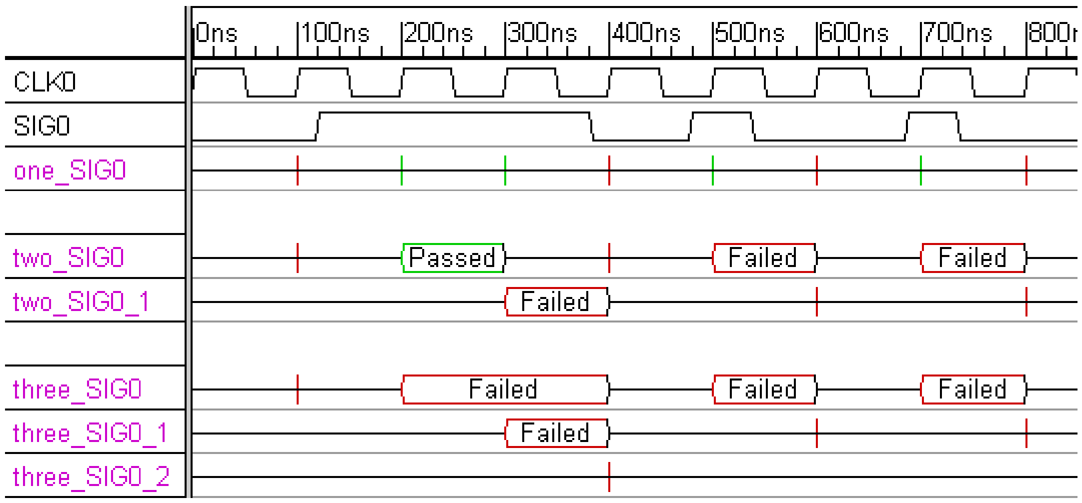
Chapter 9: Compare and Transaction Tracker Options

Waveform comparison is an optional feature and all of the SynaptiCAD timing diagram editors and graphical products can be upgraded to include these features. The Waveform Comparison feature graphically displays the differences between waveforms for two timing diagrams or individual signals. This feature is especially useful when comparing two different simulation runs, as well as for comparing logic analyzer data to a simulation run. Differences are shown as red regions on the compare signal. There are settings for tolerance regions and clocked compares to help filter out unimportant differences.


Name color warns that differences exist
Red Marks show differences
Perform Compare
Move between Differences
WaveFormer Pro
File Import/Export Edit Bus ParameterLibs Project Report View Options Window Help
Search
Verilog
SET ALL
HIDE
Diagram - analyzerData.txt*
Add Signal Add Bus Add Clock Add Spacer
Delay Setup Sample Hold Text Marker
HIGH LOW TRI VAL INVal WHI WLO HEX
View Class
30.14ns 2.688ns
0ns 2ns 4ns 6ns 8ns 10ns 12ns 14ns 16ns
0 Test.pin1
1 Test.pin1
2 Test.pin2
3 Test.pin2
4 Test.pin3
5 Test.pin3
Report - Differences
Differences data also written to a TXT for automated compares
Hyper-linked list of differences
simulation.log waveperl.log Errors Differences Grep TE_parse.log TE Results my_analyzerData_vunit.psl
Vari localhost Connected Simulation Good INS Ln: 0 Col: 0
Edit all Compare Properties like Tolerance and Clocked Compare

	Signal Name	Start Time	End Time	Reference...	Compare State
1	Test.pin3	2.5	3	0	1
2	Test.pin2	4	4.5	1	0
3	Test.pin3	6.5	8	1	0
4	Test.pin2	9	10.5	0	1

Transaction Tracker can be purchased as a separate product or as an option for any of the timing diagram editors or simulation products. Transaction Tracker is a new type of design verification tool for viewing simulation data as higher-level transactions, instead of as simple waveforms. Users specify transaction patterns (temporal assertions) to match against using the PSL Sugar language, and Transaction Tracker displays matches and partial matches of these patterns graphically as "transaction records". The matching process is similar to the matching of words using "regular expressions", but instead of matching against characters, a temporal assertion matches against the values of signals over time.



9.1 Performing a Signal Compare

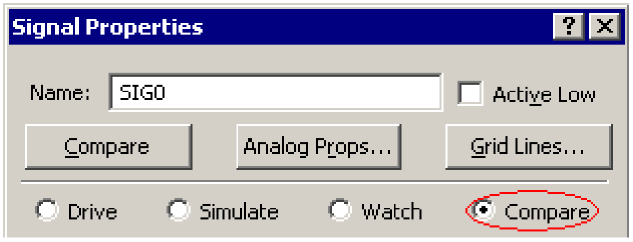
When comparing two waveforms, the **signal names must match** and one of the signals must be marked as **compare** in its *Signal Properties* dialog. When comparing two entire files, these options will be set automatically. However, if you are comparing individual signals, you will need to set these by hand. Comparing files also be done automatically using the batch mode feature discussed in Section 11.5 Batch Mode 159.

Comparing whole timing diagrams or simulation files:

- Load the first timing diagram (or other file) with the **File > Open Timing Diagram...** menu.
- Then load the second timing diagram using the **File > Compare Timing Diagram...** menu. This second file must be a timing diagram file with a btim extension, so you may need to first load this file normally and save it as a btim file prior to performing the comparison. Closing this dialog (1) loads the second set of signals into the first timing diagram, (2) sets their signal type to **compare**, and (3) performs the first compare between any two signals that have matching names.
- Normally the compare signals are arranged so that they appear directly below the original signal. However, if you uncheck the **View > Compare and Merge > Interleave Compare Signals** menu, the compare signals will be displayed as a group below all of the original signals.

Setting up a compare between two signals:

- Create two signals with the ***same name***.
- Double click on the signal name that will display the red compare regions to open the *Signal Properties* dialog, then select the **compare** radio button. This will perform the first compare and also expose additional compare settings.


Signal Properties

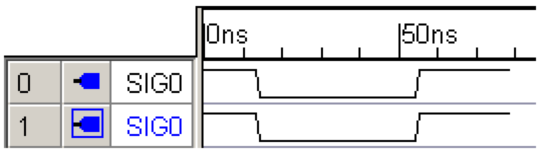
Name: SIG0

Active Low

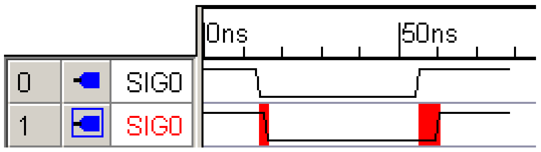
Drive Simulate Watch [✓ Compare]

The Compare Display:

- If there are no differences within the tolerance range, then the compare signal will have a **blue** name and the waveform will be all black. The direction icon will also have a box around it to indicate that it is a compare signal.

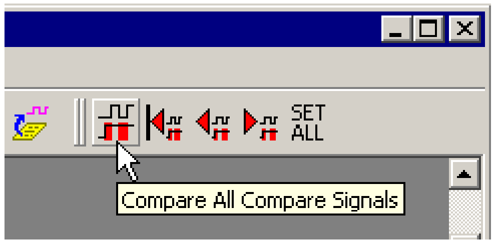


- If there are differences, then the compare signal will have a **red** name and the waveform will have red highlighted regions showing the differences.



Forcing a comparison:

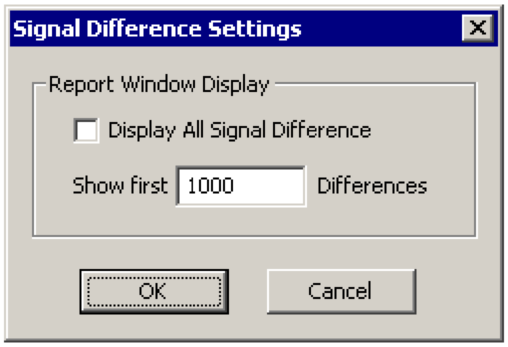
- The comparison regions are calculated once when either the compare file is loaded or when a signal changes type to Compare. To force an additional compare, press the **Compare All Compare Signals** button on the main button bar.



Control the number of differences displayed:

To avoid slowdowns in the comparison calculation when there are an unexpected number of differences, the compare feature by default will only show the first 1000 differences. To change this default:

- Choose the **View > Compare and Merge > Signal Difference Settings** menu to open the dialog.
- Either type in the number of differences to display or check the **Display All Signal Differences** box.


Signal Difference Settings

Report Window Display

☐ Display All Signal Difference

Show first 1000 Differences

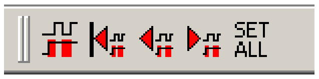
OK Cancel

9.2 Viewing Compare Differences

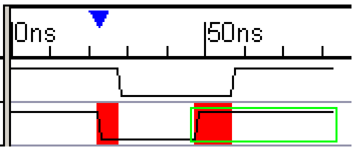
There are three ways to examine the differences that are reported; (1) graphically navigate the waveform window using the Compare tool bar, (2) double click on a difference in the Report window, and (3) view a tab delimited text file that is saved in the same directory as the diagram used for comparison.

(1) Use the Compare buttons to find the differences:

- The compare tool bar is located on the main window button bar.

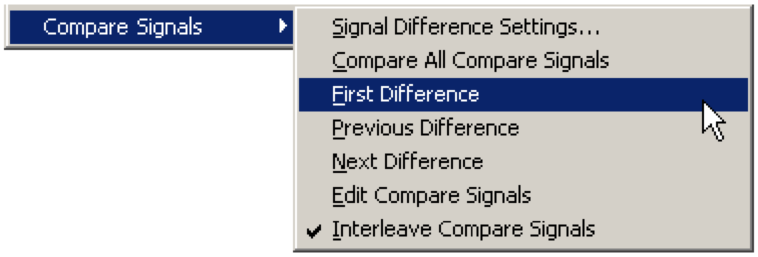


- The **Next Difference** button, finds the next difference in the timing diagram (in time), and highlights the region in green. If necessary the diagram will scroll to display the difference region.



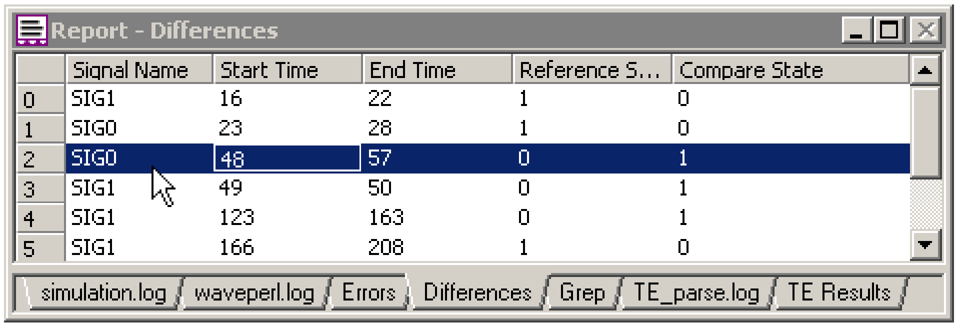
- The **Previous Difference** button finds the difference immediately before the currently highlighted difference.
- The **First Difference** button finds the difference closest to time zero.

These functions are also available in from the View > Compare Signals menu.



(2) Double click on a row in the Report Window Differences Tab

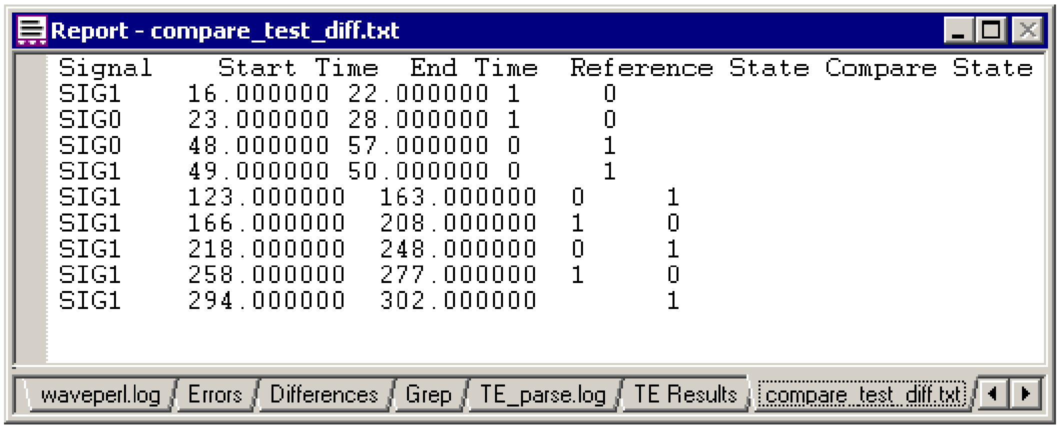
- In the *Report* window, press the **Differences** tab to display the compare differences in tabular form. If your *Report* window is not visible, select the **Windows > Report Window** menu option to bring the window to the foreground.
- Double click on a difference to highlight it in the *Diagram* window.


Report - Differences

	Signal Name	Start Time	End Time	Reference S...	Compare State
0	SIG1	16	22	1	0
1	SIG0	23	28	1	0
2	SIG0	48	57	0	1
3	SIG1	49	50	0	1
4	SIG1	123	163	0	1
5	SIG1	166	208	1	0

(3) The Differences File

The data displayed in the *Differences* tab of the *Report* window is also written to a tab-delimited text file named *TimingDiagramName*_diff.txt. This data can be used by external programs to do batch processing on the compares. You can use the *Report* window to view this file by selecting the **Report > Open Report Tab** menu item to open the file.



```
Signal    Start Time  End Time   Reference State Compare State
SIG1    16.000000 22.000000 1     0
SIG0    23.000000 28.000000 1     0
SIG0    48.000000 57.000000 0     1
SIG1    49.000000 50.000000 0     1
SIG1    123.000000   163.000000   0      1
SIG1    166.000000   208.000000   1      0
SIG1    218.000000   248.000000   0      1
SIG1    258.000000   277.000000   1      0
SIG1    294.000000   302.000000          1
```

9.3 Tolerance, Clocked, and Don't Care

The tolerance, clock, and don't care regions allow the compare feature to ignore differences that do not matter. The tolerance settings specify regions around the reference signal in which to ignore any differences. The clocked feature will cause the compare to only be performed in the specified region around the clock edges. And putting don't care regions on the reference signal, removes those regions from the compare calculation.

Open the Signal Properties dialog to set Tolerance or specify a clocking signal:

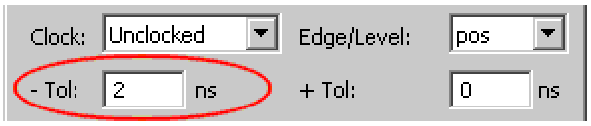
- To modify the tolerance ***for all the compare signals*** press the **Set All** button, to select the compare signals and open the *Signal Properties* dialog in group editing mode. SET ALL

- To modify the tolerance ***for one compare signal***, just double click on the signal's name to open the *Signal Properties* dialog in individual mode.

Set the Tolerance:

By default, all compare signals have a tolerance of +/- zero, so that any difference between the signals will be displayed. However, positive and negative tolerances can be specified so that the compare will ignore small differences within the range of tolerance values.

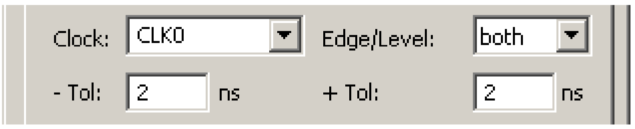
- For unclocked signals, the **-Tol** and **+Tol** boxes specify the region to *IGNORE differences* before and after the edge on the reference signal.

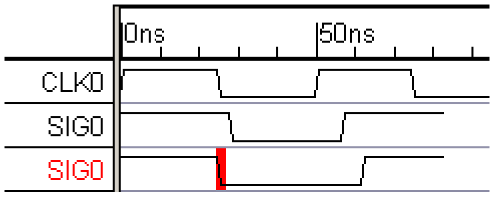

Clock: Unclocked Edge/Level: pos
- Tol: 2 ns + Tol: 0 ns

Setup a Clocking Signal:

Clocked comparisons provide the ability to compare signals on clocked edges. This is useful if you want to verify that signal values match after state transitions in a clocked system, but are not concerned with the signal values between those transitions (e.g. to ignore "glitch" differences).

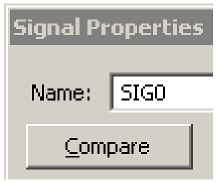
- Select the clocking signal from the **Clock** box and indicate which edges of the clock signal will be used for the time of comparison using the **Edge/ Level** box.
- For clocked signals, the **-Tol** and **+Tol** boxes specify the region in which to *MARK the differences* before and after the clock edge. (This is the opposite meaning from the unclocked signal).


Clock: CLK0
Edge/Level: both
- Tol: 2 ns
+ Tol: 2 ns



Force a compare using the Signal Properties dialog

- Press the **Compare** button in the *Signal Properties* dialog, or press the **Compare All Compare Signals** button to force the a compare.

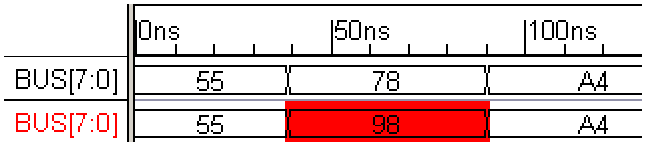


OR

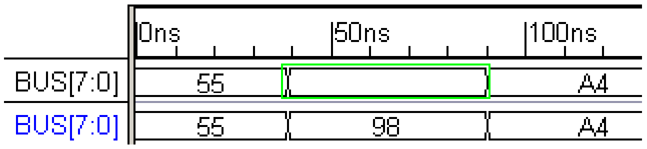
Setting Don't Care Regions

Sometimes you may not want to compare certain regions in time on a particular signal. One case would be on a bus signal where the actual data did not matter. If you want the compare function to skip a particular section, then just turn the waveform into a blank valid region on the reference signal.

- Locate the waveform section on the reference signal (not the compare signal).



- If the segment is a valid segment with a value in it, then double click on the segment to open the *Edit Bus State* dialog and erase the value in the **Virtual** box.



- If the segment is a graphical state other than valid, then highlight the section by clicking on it and then press the **VAL** waveform button to change the graphical state to valid.
- Rerun the compare function using the **Compare All Compare Signals** button.

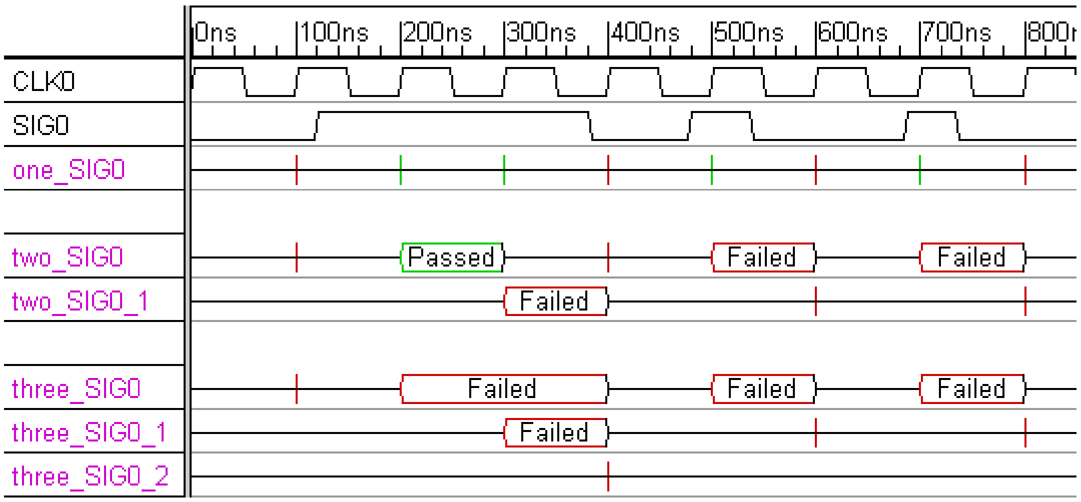
If you are creating a lot of don't care regions, then save the data in your reference file before you load the first compare file. That way each time that you bring in a new simulation run or logic analyzer capture the reference file will already be formatted properly.

9.4 Transaction Tracker

The Transaction Tracker Option displays areas in a waveform file that match a transaction pattern defined as a PSL (Sugar) equation.

Vocabulary

- **Temporal assertion**: an equation-based description of a "transaction pattern".
- **Transaction record**: an attempted match of a temporal assertion at some particular time.



Using Transaction Tracker

Generally the first step when using Transaction Tracker is to load a waveform file containing simulation results or data captured by a logic analyzer. After loading a waveform file, you can add new signals and define transaction patterns to analyze the waveform data. The transaction patterns are written as temporal assertions in the industry-standard PSL/Sugar assertion language and a clocking signal is specified that indicates when the assertion should be evaluated.

Once the temporal assertion is simulated, the associated signal will display the fails and passes of the pattern as transaction records. A new match attempt will be made each clock cycle and several match attempts can proceed in parallel. If a transaction fully matches the pattern, the transaction record will be displayed in green. If at some point a transaction fails to match the entire pattern, its transaction record will terminate at that point and display in red as a failed match.

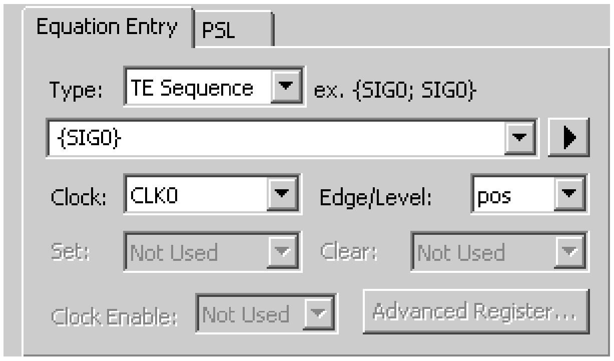
A match is attempted for a transaction pattern each new clock cycle. If the pattern does not begin to match during a cycle, a red spike will be created. If the pattern is fully matched during a single cycle, you will see a green spike at this cycle. If a pattern takes several cycles to match or fail, the transaction record will display as a green or red valid segment.

Most transaction patterns require several clock cycles to match, so it is possible that a pattern may generate several transaction records that overlap in time. Transaction Tracker will create additional "overflow signals" when these overlaps occur, so that the original signal's display does not get too cluttered. The later overlapping records will be displayed on these overflow signals (the time used to determine the placement of the transaction records is the "end" of the transaction, not the beginning).

To create an assertion, perform the following steps:

- Select **File > Load Timing Diagram** to load a file with waveform data to analyze. The file can be a timing diagram, a VCD file, or any other waveform file supported by SynaptiCAD.
- Create a new signal by clicking on the **Signal** button, then ***double click*** on the signal name to open the *Signal Properties* dialog.

- In the **Equation Entry** tab, set the **Type** drop-down to either **Temporal Sequence** or **Temporal Property**.
- Type a temporal assertion (e.g. {SIG0}) into the edit box below the equation type drop-down.
- From the **Clock** drop-down, choose a clock signal for the assertion and set the **Edge Level** setting to either **pos** for rising edge sampling or **neg** for falling edge sampling.



- Click on the **Simulate** radio button to activate the assertion for simulation.

Frequent Usage Tip:

- Use the **File > Merge Timing Diagram** to merge in a file containing your transaction tracker signals. After you analyze a simulation file for the first time, save off the transaction signals to a separate diagram via copy and paste, so that you do not have to re-create your transaction signals containing the temporal assertions when you want to repeat the analysis on a different set of waveform data.

Syntax for Transaction Tracker:

Transaction Tracker supports the "simple subset" of PSL version 1.1. The simple subset is the part of PSL supportable by simulators; the remaining PSL functionality is primarily intended for formal verification tools. To get a quick introduction, we recommend you start by launching Transaction Tracker (or other SynaptiCAD tool with the Tracker option), loading the psltutorial1.btim timing diagram file located in **C:\SynaptiCAD\Examples\sugar**, and reading the associated **Transaction Tracker Tutorial** in the online help. A valid license is required to perform this tutorial.

After reading the introductory tutorial above, we also recommend that you read the general PSL Sugar tutorial found in **c:\SynaptiCAD\Examples\sugar\sugar_tutorial.pdf**.

For a complete copy of the PSL 1.1 Language Reference Manual (LRM), please visit http://www.haifa.il.ibm.com/projects/verification/sugar/index.html. This site is maintained by the creators of the Sugar assertion language, and it also contains a number of other documents discussing various aspects of PSL/Sugar that may be of interest.

Temporal Assertion Error Reporting Procedence

1) When defining an assertion using the *Signal Properties dialog*, the dialog checks if a clock has been specified in the **Clock** drop down and generates an error message box if not.

2) Next, the PSL assertion parser checks for illegal PSL syntax. PSL syntax errors are reported in the **TE_parse.log** tab in the Report window. Note: the parser does NOT verify that the signal names in the assertion are valid signals, as the signals will generally be declared in the attached verilog code (this check is made in the next step below).

3) If the above checks succeed, the Verilog compiler will execute and report any references to unknown identifiers in PSL assertions (because an associated verilog file is generated for the PSL file). These will be reported as an "undeclared signal error in diagram_name_vunit.v" typically. Errors detected by the Verilog compiler will be reported in the Errors tab of the Report window and in the simulation.log file.

Files Generated by Transaction Tracker

During a transaction tracker simulation run, the following files will be generated, given an input diagram of diagram_name.btim:

- ***diagram_name*_vunit.psl:** PSL file containing PSL code for the assertions in the btim file. The assertions are grouped into vunits that are clocked off the same clock edge.
- ***diagram_name*_vunit.v:** Verilog file that connects PSL equations to the HDL source code.
- ***diagram_name*Tim.v**: Verilog file that contains stimulus signals and HDL simulated signals.
- **TE_parse.log:** Error messages generated during parse of PSL assertions.
- **Simulation.log:** Error messages generated during compile of HDL code and log messages generated during simulation run.
- ***diagram_name*_pslresults.txt:** Text file containing the transaction records of all the PSL assertions, in time order.

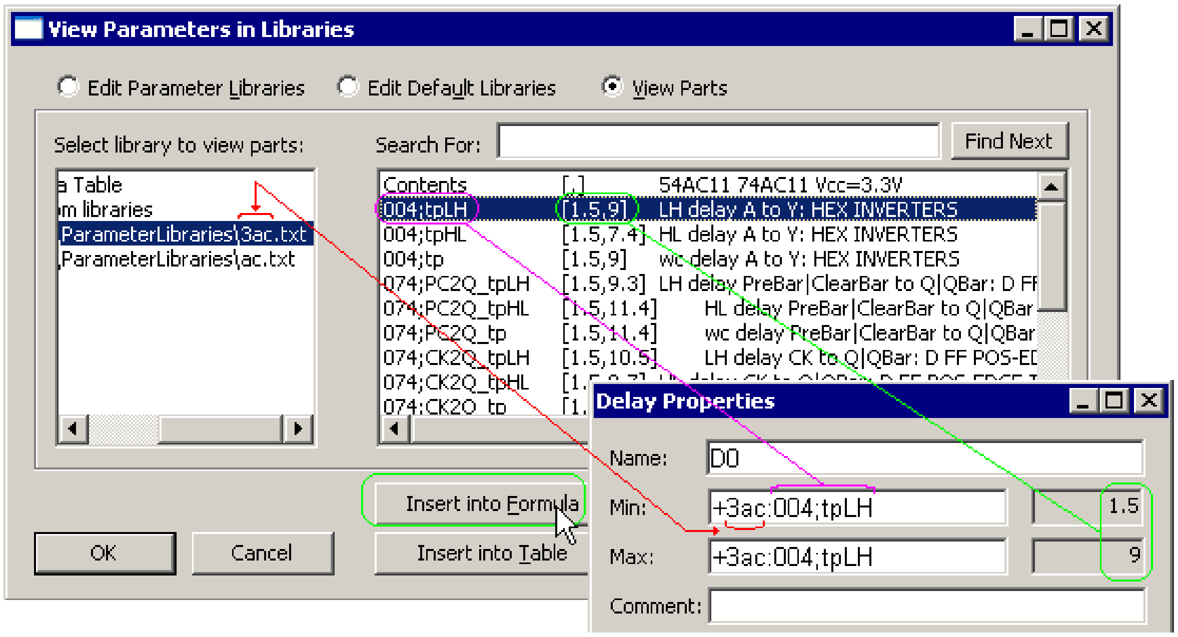
This file is created by selecting the menu option Import/Export > Export and Display Temporal Expression Results. The same results will also be displayed in the TE Results tab in the Report window.

Notes and Limitations

- TE signals can reference TE signals and Boolean Equations, but Boolean equations cannot reference TE signals.
- Transaction Tracker supports only the vunit and assert commands from the verification layer. The modeling layer is not applicable to Transaction Tracker.
- The following PSL built-in functions are not yet supported: isunknown, countones, onehot, onehot0, next, prev, stable, and the union operator.
- PSL endpoints are not yet supported.

Chapter 10: Parameter Libraries and Merging Diagrams

Parameter libraries are free parameter files that can be shared by several timing diagram projects. They 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 on-line component information (see T11.13 TDML Support 176). You can also use the timing diagram editor to create your own libraries.



```
Contents        [,]        54AC11 74AC11 Vcc=3.3V
004;tpLH        [1.5,9]    LH delay A to Y: HEX INVERTERS
004;tpHL        [1.5,7.4]  HL delay A to Y: HEX INVERTERS
004;tp          [1.5,9]    wc delay A to Y: HEX INVERTERS
074;PC2Q_tpLH   [1.5,9.3]  LH delay PreBar|ClearBar to Q|QBar: D FF
074;PC2Q_tpHL   [1.5,11.4]     HL delay PreBar|ClearBar to Q|QBar
074;PC2Q_tp     [1.5,11.4]     wc delay PreBar|ClearBar to Q|QBar
074;CK2Q_tpLH   [1.5,10.5]     LH delay CK to Q|QBar: D FF POS-ED
```

Name: D0
Min: +3ac:004;tpLH 1.5
Max: +3ac:004;tpLH 9

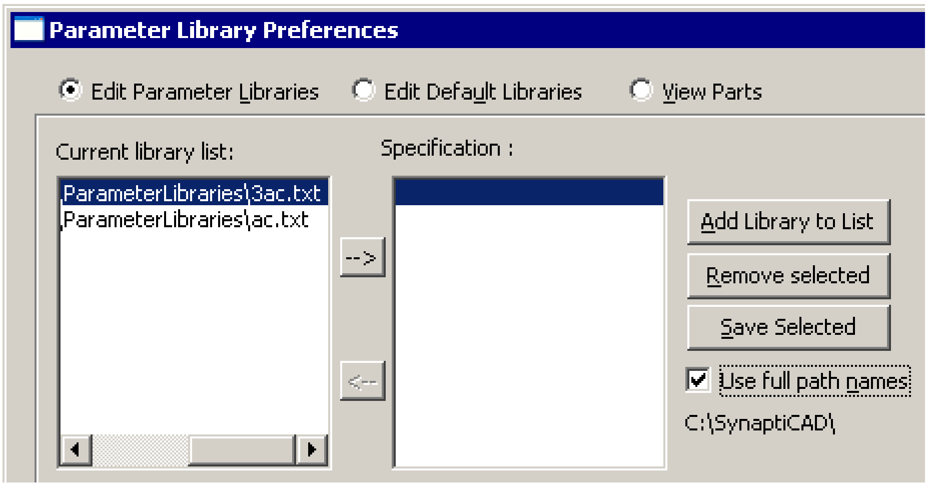
Diagrams can also be merged into one another so that waveforms and parameters are visible in one diagram. When merging diagrams parameter conflicts are displayed in a dialog and can be resolved before the merge begins.

10.1 Adding Libraries, Specifications, and Macros

Before you can use a parameter library in a timing diagram project you must (1) add the library to the library search list, (2) set up a specification, and (3) define a macro for the library.

1) Add parameter libraries to your Library Search List:

- Select the **ParameterLibs > Parameter Library Preferences** menu option to open the *Parameter Library Preferences* dialog. Then choose either the **Edit Parameter Libraries** radio button to add libraries to the current diagram, or the **Edit Default Libraries** radio button to add them to default list for all new timing diagrams.

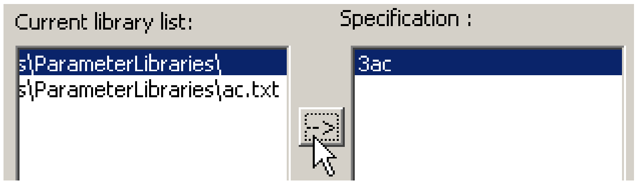


- Click the **Add Library to List** button to open a file dialog, then select the names of the parameter libraries that you want to add to the library list and press the Ok button to close the file dialog.
- Notice that the **Use full pathnames** checkbox is checked. This means that both the library 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 libraries you selected are now listed in the **Current library list**. At this point you can reference and use the values in the libraries. DO NOT close the *Library Preferences* dialog as it is used in the next step (this dialog is modeless, so you can leave it open while you perform regular operations).

2) Define specifications for the parameter libraries in the Library Search List (optional, but recommended):

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 specifications in order to make parameter searches faster.

- In the *Parameter Library Preferences* dialog, select a library and press 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:

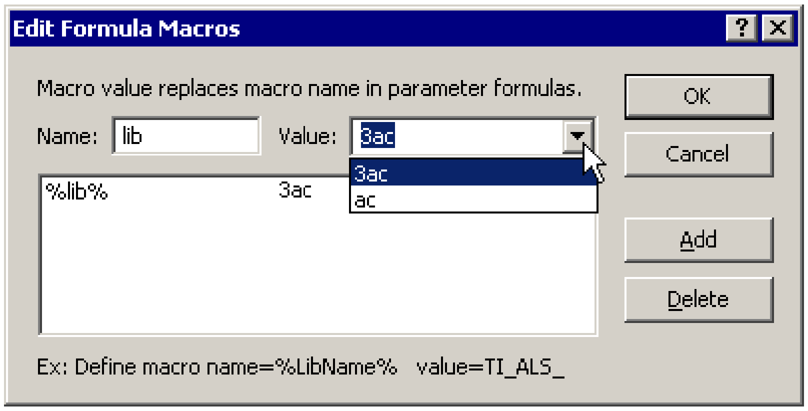
  LibrarySpecification*:*ParameterName

- The next section, 10.2 Referencing Library parameters, shows how to reference a library parameter without having to type the name and specification. Click **OK** to close the dialog.

3) Define a Macro to represent the specifications of the new parameter library (optional):

Macros allow you to substitute one string or character for another in formula evaluation. Inside of a formula, a macro name is surrounded by percentage signs to indicate that it is a macro. 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.

- Select the **ParameterLibs > Macro Substitution List** menu option to open the *Edit Formula Macro* dialog



Macro value replaces macro name in parameter formulas.

Name: lib Value: 3ac

3ac
ac

%lib% 3ac

Ex: Define macro name=%LibName% value=TI_ALS_

- Enter the macro name into the **Name** box and the string that you want to replace in the **Value** box, then press the **Add** button. Notice that all of the library specifications are listed in the Value drop-down, so you do not have to re-type them. Percentage characters are not allowed in macro names.
- The above image defines a macro called lib. When a parameter in a formula is specified as ***%lib%partname***, the %lib% part will be replaced with **3ac.partname**. Later if you want to evaluate the timing using the ac library, you can come back to the *Edit Formula Marcos* dialog and change the value of the macro to ac and the entire timing diagram will change. 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.
- Any number of macros may be used in a single string. However, nesting of macros is not allowed.

-

10.2 Referencing and Viewing Library Parameters

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. The parameter names can be typed into a min/max time formula, or they can be automatically inserted by using the *View Parameters in Libraries* dialog.

Syntax for Referencing a parameter located in a library:

- 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**.

```
LibrarySpecification:ParameterName

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**).

```
ParameterName

00;tpLH
```

Automatically insert a the parameter into a formula (no typing needed):

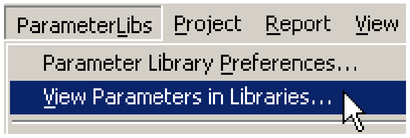
- Open the *Parameter Properties* dialog, by double-clicking on a delay, setup, or hold, and leave the dialog open.
- Also open the ***View Parameters in Libraries*** dialog, by either:

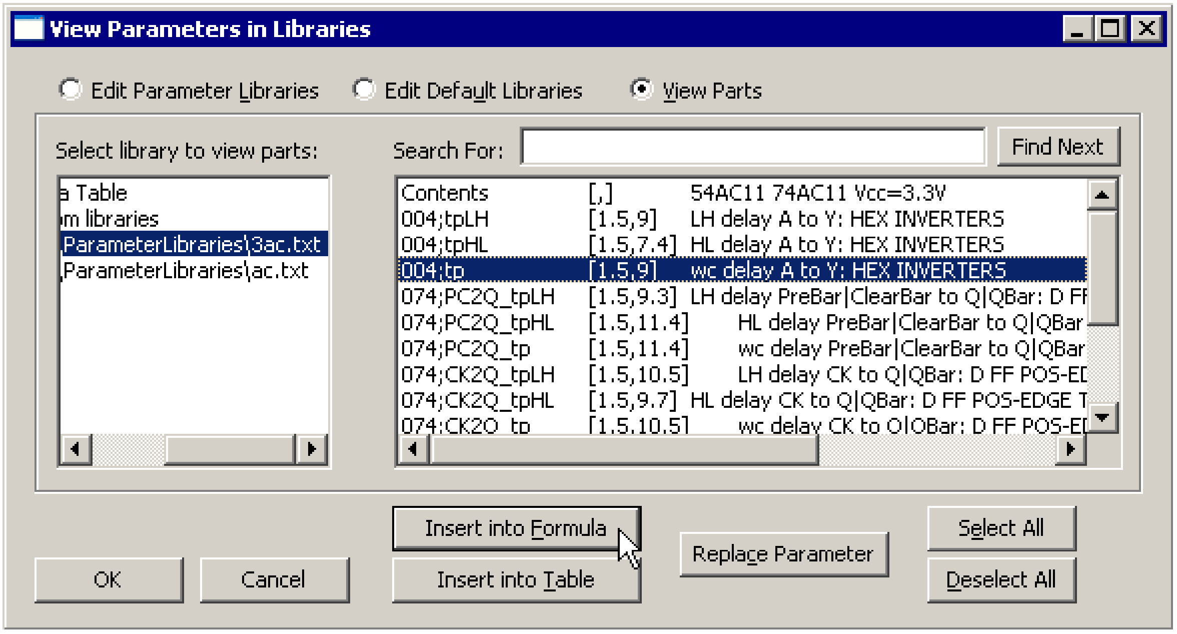
- In the *Properties* dialog, pressing the **Libraries** button (this will edit both min and max formulas).

- OR, in the *Properties* dialog, clicking inside the min or max box and then pressing the **<F3>** (this will edit only the box selected)

<F3>

- OR, choosing the **ParameterLibs > View Parameters in Libraries** menu.


ParameterLibs
Project
Report
View
Parameter Library Preferences...
View Parameters in Libraries...



Edit Parameter Libraries   Edit Default Libraries   [✓ View Parts]

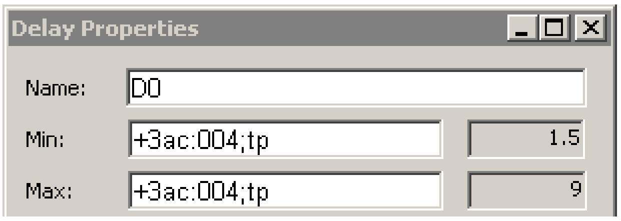
```
a Table
m libraries
ParameterLibraries\3ac.txt
ParameterLibraries\ac.txt
```

```
Contents         [,]         54AC11 74AC11 Vcc=3.3V
004;tpLH         [1.5,9]     LH delay A to Y: HEX INVERTERS
004;tpHL         [1.5,7.4]   HL delay A to Y: HEX INVERTERS
004;tp           [1.5,9]     wc delay A to Y: HEX INVERTERS
074;PC2Q_tpLH    [1.5,9.3]   LH delay PreBar|ClearBar to Q|QBar: D FF
074;PC2Q_tpHL    [1.5,11.4]      HL delay PreBar|ClearBar to Q|QBar
074;PC2Q_tp      [1.5,11.4]      wc delay PreBar|ClearBar to Q|QBar
074;CK2Q_tpLH    [1.5,10.5]      LH delay CK to Q|QBar: D FF POS-ED
074;CK2Q_tpHL    [1.5,9.7]   HL delay CK to Q|QBar: D FF POS-EDGE T
074;CK2Q_tp      [1.5,10.5]      wc delay CK to Q|QBar: D FF POS-ED
```

- Select a library to display the parts in the library, then select a part to reference. The **Search For** box can be used find a parameter in the list.
- Next insert or replace the parameter by pressing the appropriate button.

Copyright © 2011, SynaptiCAD Sales, Inc.

- Pressing the **OK** button or the **Insert into Formula** button copies +*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, but it can be removed or changed if necessary. This is the most common method to reference the library parameter



- The **Replace Parameter** button overwrites the ***name*** and ***comment*** in addition to the min and max values displayed in the *Parameter Properties* dialog.
- The **Insert into Table** button creates a new free parameter in the Parameter Table that 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.

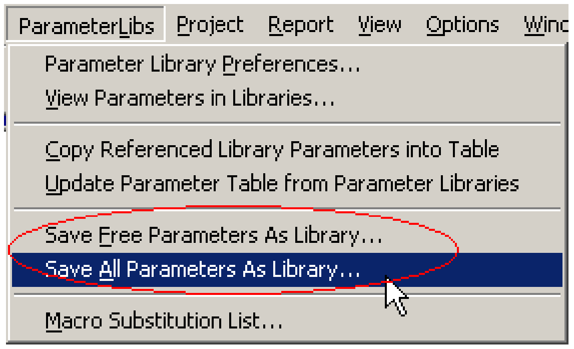
10.3 Making Parameter Libraries

Parameter libraries can be made using the timing diagram editor or they can be imported using a commercial spreadsheet. Very large libraries should be made using a commercial spreadsheet because they have more viewing and editing features than the built-in Parameter table. Library files can be stored in three different 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. Stores Base time unit 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 or *.btim):** 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.

To make a parameter library using the timing diagram editor:

- Open a timing diagram and add free parameters to the Parameter table (see section 5.6 Parameter Table 85) or delays, setups, or holds to the diagram.
- From the **ParameterLibs** menu select one of the Save Library options depending on which parameters you want to save to the file. This will open the *Save As* dialog box


ParameterLibs
Project
Report
View
Options
Winc
Parameter Library Preferences...
View Parameters in Libraries...
Copy Referenced Library Parameters into Table
Update Parameter Table from Parameter Libraries
Save Free Parameters As Library...
Save All Parameters As Library...
Macro Substitution List...

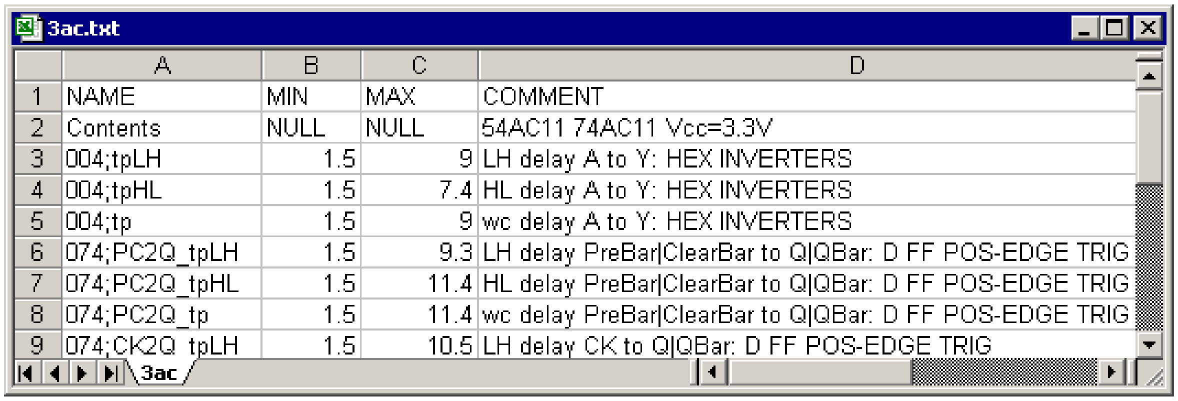
- Pick one of the free parameter formats from the **Save as type** box choose. The **txt** format is recommended for faster loading. Press the **Save** button to generate the file.

```
Free Param Text (*.txt)
Free Parameter (*.fp)
```

- Once the library is made it can be edited just like any other timing.

To make a parameter library using a commercial spreadsheet:

- Open a commercial spreadsheet program like Microsoft Excel.
- 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. The min and max columns can accept either time values or time formulas as described in section 2.5 Time Formulas for Clocks and Parameters 47.


3ac.txt

	A	B	C	D
1	NAME	MIN	MAX	COMMENT
2	Contents	NULL	NULL	54AC11 74AC11 Vcc=3.3V
3	004;tpLH	1.5	9	LH delay A to Y: HEX INVERTERS
4	004;tpHL	1.5	7.4	HL delay A to Y: HEX INVERTERS
5	004;tp	1.5	9	wc delay A to Y: HEX INVERTERS
6	074;PC2Q_tpLH	1.5	9.3	LH delay PreBar|ClearBar to Q|QBar: D FF POS-EDGE TRIG
7	074;PC2Q_tpHL	1.5	11.4	HL delay PreBar|ClearBar to Q|QBar: D FF POS-EDGE TRIG
8	074;PC2Q_tp	1.5	11.4	wc delay PreBar|ClearBar to Q|QBar: D FF POS-EDGE TRIG
9	074;CK2Q_tpLH	1.5	10.5	LH delay CK to Q|QBar: D FF POS-EDGE TRIG

- Use the **Import/Export** or **Save As** menu 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.

Converting from TXT to FP format:

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 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.

- Select the **File > New Timing Diagram** menu to make sure that no project is currently loaded.
- Set the **Options > Display Time Unit** menu to match that of the library you are about to import.
- Select the **File > Open Timing Diagram** menu and open the file that you want to convert.
- Look at the *Parameter* window and make sure that the parameter values are correct.
- Next, save the library by using the **ParameterLibs > 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.

10.4 Merging Diagrams

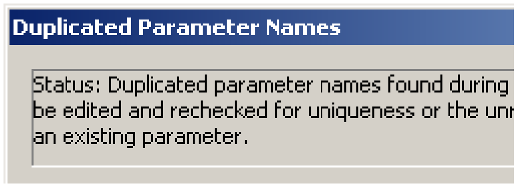
You can merge waveforms and parameters from another diagram into the currently active timing diagram. If the two diagrams each have a parameter with the same name but different properties (min, max, margin, comment) or a different type (delay, setup, hold), then a dialog will open so that you can resolve the conflict. This dialog will also open if a conflict occurs during a *Search and Rename* Parameter operation (see Section 5.6 Parameter Window 85). The Options menu also contains a properties dialog that specifies how signals are merged into the diagram and how parameter conflicts can be automatically fixed.

Merging Diagrams:

- Open the timing diagram that will hold all of the merged signals, using **File > Open Timing Diagram** menu.
- Select **File > Merge Timing Diagram** menu and choose the second timing diagram file. This will copy the entire diagram and paste it into the first diagram. If duplicate parameters are detected, then see below for resolving the conflicts. The merging diagram's library parameters are not available, so perform that steps in 10.5 Self-Contained Timing Diagrams 146 to copy the library parameters into the diagram before merging.

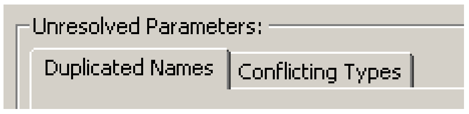
Resolving Duplicate Parameters on a diagram merge or parameter rename:

- The *Duplicated Parameter Names* dialog opens when a name or type conflict is found during a diagram merge or a parameter *Search and Rename* operation. Conflicts can be resolved by either renaming to create a new parameter or by converting to an existing parameter. If this dialog does not open on conflicts, then the Merge property **Prompt on Parameter Name Duplication** may be disabled (see next subtopic).
- The Status box gives updated information on each parameter merge and rename operation and whether it was successful, so keep checking this after you attempt to resolve conflicts.


Duplicated Parameter Names

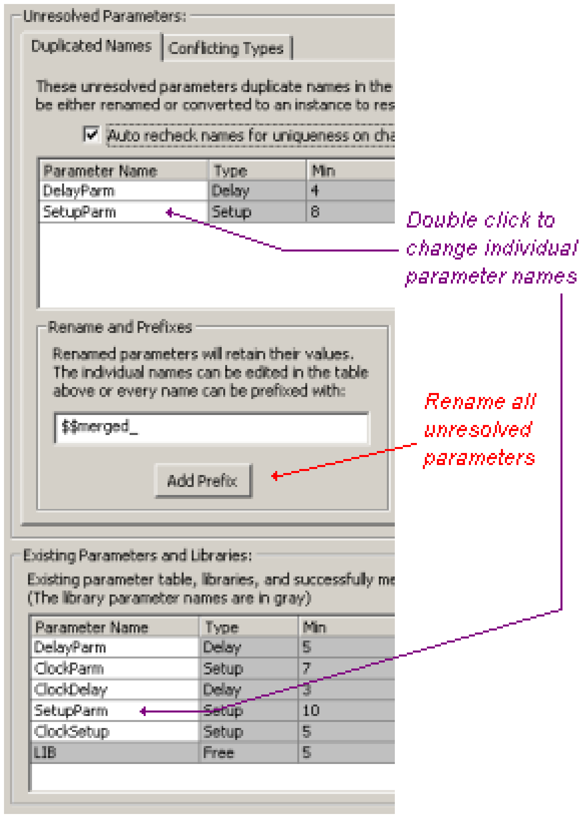
Status: Duplicated parameter names found during
be edited and rechecked for uniqueness or the unr
an existing parameter.

- The **Duplicate Names** Tab shows parameters that have the same name and type, but differ only in properties like min or max times.



- The **Conflicting Types** Tab shows parameters that have the same name but different parameter types (e.g. one is a delay and one is a hold).

- One way to resolve a conflict is to change the name of the parameter. Double click on the parameter name to open an edit box. The **unresolved parameters** list contains the parameters from either the diagram specified in the File Merge operation, or the parameter which is being renamed. The **existing parameters** list contains parameters from the original diagram or which are not being renamed.
- All the unresolved parameters can be renamed as a group by pressing the **Add Prefix** button. This inserts the specified prefixed in front of all the unresolved parameter names.
- The Existing Parameters and Libraries List contains the current diagram's parameters and libraries, and the merging diagram's parameters that have unique names or have been successfully merged or renamed. The merging diagram's library parameters are not available, so perform that steps in 10.5 Self-Contained Timing Diagrams 146 before merging.


Unresolved Parameters:

Duplicated Names | Conflicting Types

These unresolved parameters duplicate names in the
be either renamed or converted to an instance to res

☑ Auto recheck names for uniqueness on cha

Parameter Name	Type	Min
DelayParm	Delay	4
SetupParm	Setup	8

Rename and Prefixes

Renamed parameters will retain their values. The individual names can be edited in the table above or every name can be prefixed with:

$$merged_

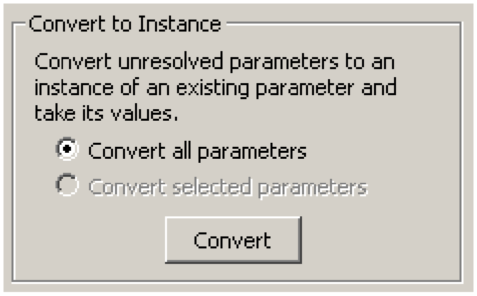
Existing Parameters and Libraries:

Existing parameter table, libraries, and successfully me
(The library parameter names are in gray)

Parameter Name	Type	Min
DelayParm	Delay	5
ClockParm	Setup	7
ClockDelay	Delay	3
SetupParm	Setup	10
ClockSetup	Setup	5
LIB	Free	5

Add Prefix
Double click to change individual parameter names
Rename all unresolved parameters

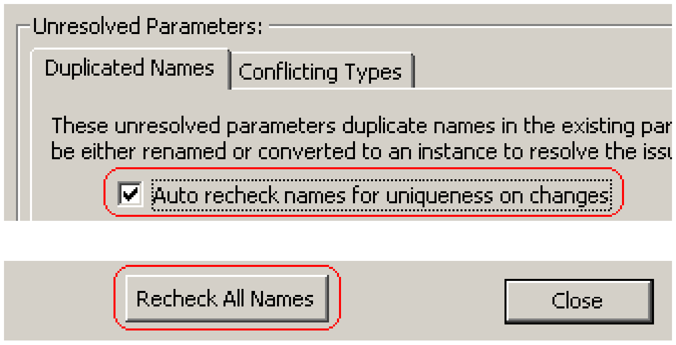
- Another way to resolve a conflict is to convert the parameter being merged into a parameter that exists in the diagram. The merged parameter will lose its properties and take the properties of the existing parameter. To convert one parameter, select it in the existing list and press the **Convert** button.


Convert to Instance

Convert unresolved parameters to an instance of an existing parameter and take its values.

[✓ Convert all parameters]
Convert selected parameters

- With the **Auto Recheck** option enabled, any changes to the unresolved parameter names are immediately checked against the existing parameter name and merged, if the check is successful. This does not check name changes made to existing parameters, so use the **Recheck All Names** button to check these changes.


Unresolved Parameters:

Duplicated Names | Conflicting Types

These unresolved parameters duplicate names in the existing par
be either renamed or converted to an instance to resolve the issu

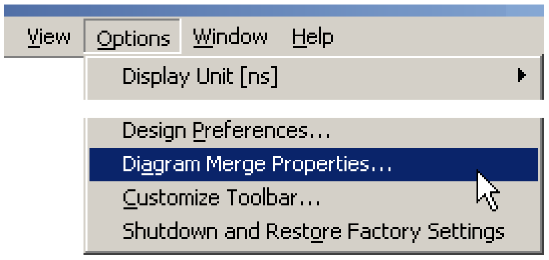
☑ Auto recheck names for uniqueness on changes

- If you press **Close** button before all the conflicts have been resolved, the program will attempt to preserve the data by changing the unresolved parameter names.

Merging Properties

The Diagram Merge Properties dialog controls how signals are grouped after the merging of two diagrams. It also controls how duplicate parameter errors are handled.

- Choose the **Options > Diagram Merge Properties** to option the dialog



- When **Interleave Signals on Merge** is enabled, signals with identical names will be grouped together after a merge operation.
- If **Prompt on Parameter Name Duplication** is enabled, then the *Duplicated Parameter Names* dialog will open when a name or type conflict is found during a diagram merge operation. If it is unchecked, then the program will automatically fix duplicate parameters using the two settings below the control. This function is always disabled during a batch merge.

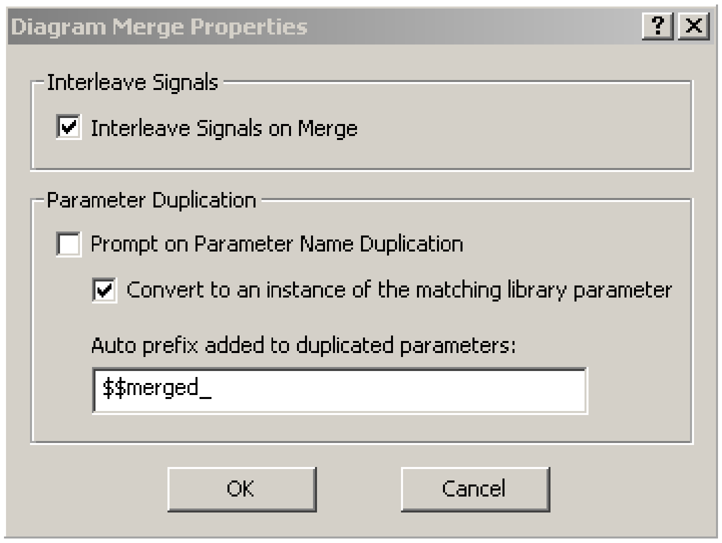


Diagram Merge Properties

Interleave Signals

[✓] Interleave Signals on Merge

Parameter Duplication

[ ] Prompt on Parameter Name Duplication

[✓] Convert to an instance of the matching library parameter

Auto prefix added to duplicated parameters: $$merged_

- If **Convert to an instance of a library parameter**"is enabled, the merging diagram's parameters will take the values of any matching parameters from the current diagram's libraries. If unchecked, the duplicated parameters will get the stated prefix.
- During an automated merge, duplicate parameters in the merging diagram will be renamed by adding the specified prefix string to the parameter name.

10.5 Self-Contained Timing Diagrams

During the design process, projects usually reference free parameters contained in parameter libraries. But when it is time to archive the timing diagrams, it is inconvenient to have to carry around all the libraries with your diagrams. Also you may not want the parameters updated in the timing diagram, if at a later time the libraries are updated. In these cases it is generally preferable to copy all the library parameters that are referenced into the timing diagram itself. That way the BTIM file contains all the free parameters it needs to draw itself.

Copy Library Parameters into the timing diagram to make a self-contained project:

- Select the **ParameterLibs > Copy Referenced Library Parameters into Table** menu to copies every library free parameter that is ***referenced*** into the timing parameter table.

- A message will appear asking if you want to hide the free parameters when you copy them into the diagram.
- Also if you are planning to move your timing diagram, then you should *empty the parameter library list* to insure that the program will not attempt to read library files not available at the new location (see section 10.1 Adding Libraries 138). If you are not moving your timing diagra, then leaving the library list intact will not affect the diagram because parameters in the project take precedence over those in the libraries.

Merge an entire Library into the timing diagram to make a self-contained project:

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

- Open the timing diagram file, then select the **File > Merge Timing Diagram** menu option to open the *File* dialog. See Section 10.4 Merging Diagrams 144 for more information on resolving parameter name conflicts that might occur during the merge.
- Select the name of the library file and choose OK.
- Note: Merging should be only be used with small libraries, because the parameters fill up the parameter table and make things hard to find. Parameters in the *Parameter* window can be hidden by selecting the parameter and choosing the **View > Hide Selected Parameter Row** menu.

Update all parameters with data from every library in the library list:

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.

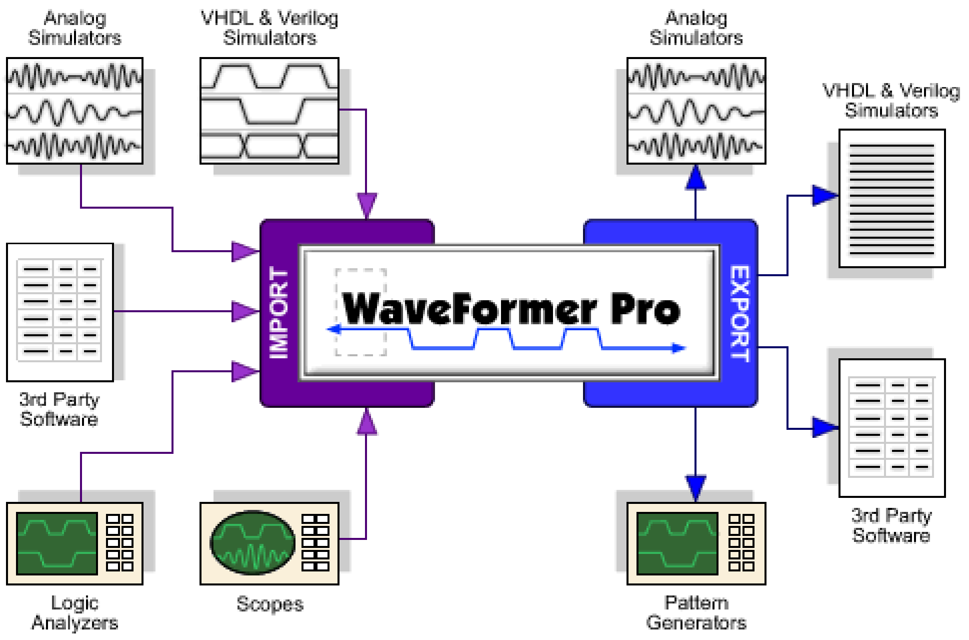
- Select the **ParameterLibs > Update Parameter Table from Parameter Libraries** menu to replace the timing data of free parameters in the project with data from library free parameters of the same name.

Update a specific free parameter with data from a certain library:

- Select the **ParameterLibs > View Parameter Library Parts** men to open the *View Parameter Library Parts* dialog.
- Select a specific parameter to update, and then press the **Insert into Table** button.

Chapter 11: Simulator and Test Equipment Support

WaveFormer Pro and DataSheet Pro can import and export waveform data from over 50 different formats including simulators, pattern generators, and logic analyzers. Many of the import and export scripts are written in Perl and can be edited or modified by the customer to work for other simulators or in-house equipment. This is covered in Appendix B: Writing Perl Scripts 191. These features are not included in Timing Diagrammer Pro and a few other products.



General background on importing and exporting files

If you are working with large files, make sure to review the information in the general sections so that you can take advantage of all of the automation features for filtering file information and using internal scripts to modify waveforms and set mappings for the test equipment pins.

- 11.1 Signal Names for export and import 149
- 11.2 Import General Instructions 150
- 11.3 Export General Instructions 154
- 11.4 Map Signals to Test Equipment Pins 157
- 11.5 Batch Mode 159

VHDL and Verilog Support

Specific Instructions for VHDL and Verilog export is located in Section 4.4 Export VHDL and Verilog testbenches 65 because the export types and simulator properties settings effect the test bench generation. The program can also read VHDL and Verilog simulator files like VCD and other formats, however these are just an easy imports covered under the 11.2 Import General instructions 150.

SPICE Support

Specific Instructions for the different SPICE imports an exports are covered in Chapter 8 Analog Display, because the *Analog Properties* dialog controls how signal data is interpret during the export and the ability to properly display the information after the input. See sections 8.5 Exporting SPICE Stimulus 124 and 8.6 Importing SPICE Stimulus 126.

Third Party File Support Sections

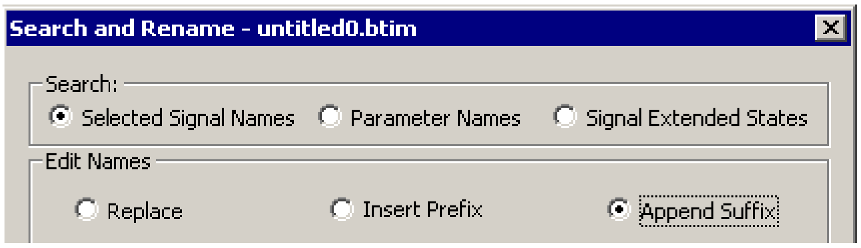
Most of the import and export formats are adequately covered in the general instructions for importing and exporting (sections 11.2 and 11.3). However, some of the test equipment formats and some of the clocked formats require a little extra work like setting up a clocking signal or defining mappings to the test equipment pins.

11.1 Signal Names for export and import

Signal names, parameter names, and bus states can be globally searched and replaced in the diagram based on a regular expression. When a file is imported from a simulator or logic analyzer, the signal names sometimes have extra information, like hierarchical model names, that need to be striped away before exporting to the next format. Also, signal names with periods are treated as if the entire name is surrounded by escape characters. This section shows examples of the regular expressions that can be used in the *Search and Rename* dialog, but more information about parameter name searching is covered in Section 5.6 Parameter Window 85, and the signal extended state search is covered in Section 3.1 Edit Virtual Bus 51.

Search and Rename Signals, Parameters, and Waveform states:

- Select the signals whose names you'd like to change, and choose the **Edit > Search and Rename** menu to open the *Search and Rename* dialog.
- Choose the type of object to search for: signal names, parameter names, or extended states.

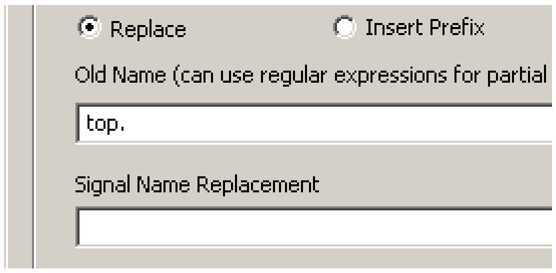


Search and Rename - untitled0.btim

Search: [✓ Selected Signal Names] Parameter Names Signal Extended States

Edit Names: Replace Insert Prefix [✓ Append Suffix]

- To strip off part of a name, select the **Replace** button, enter the search pattern into the **Old Name** box, and enter the pattern that you want to search for, and enter the replacement string in the **Signal Name Replacement** box
- In the example, "top." is being replaced by a "blank". The pattern can also match to any part of the string (not just the beginning).

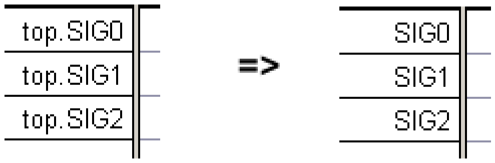


[✓ Replace] Insert Prefix

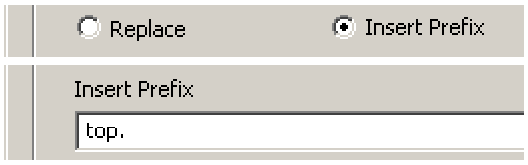
Old Name (can use regular expressions for partial

top.

Signal Name Replacement



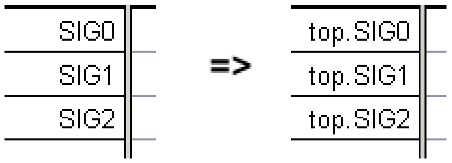
- To insert text before or after the name select either the **Insert Prefix** or the **Append Suffix** button.
- Then enter the text to add to the names in the edit box.



Replace [✓ Insert Prefix]

Insert Prefix

top.



- Sometimes during a **Parameter** rename operation the program can encounter a duplication conflict when a parameter is being renamed to the same name as another parameter, but that the two parameters differ in properties (different min and max times) or types (delay, setup, hold). If such a conflict is found the *Duplicated Parameter Names* dialog will open so that you can fix the conflict. This dialog is described in Section 10.4 Merging Diagrams 144, because this type of conflict happens frequently during a timing diagram merge.

Signal names and the hierarchy/scope character (".")

If your signal name contains one or more periods, the timing diagram editor will assume that the signal name is meant to be hierarchically scoped within the source code. For example, a signal named `Foo.Bar` will be treated as signal `Bar` in a module named `Foo`. If you want to avoid this automatic scoping, add backward slashes to the beginning and end of your signal name to 'escape' it. For example, a signal named `\Foo.Bar\` will be treated as a signal at the top level of the scope hierarchy.

If your signal name contains two or more directly adjacent periods (foo..bar), the timing diagram editor will automatically 'escape' the signal name, as there are no characters in between that could be a scope name. For example, a signal named `Foo..Bar` will be treated in the same manner as if its name were `/Foo..Bar/`.

11.2 Import General Instructions

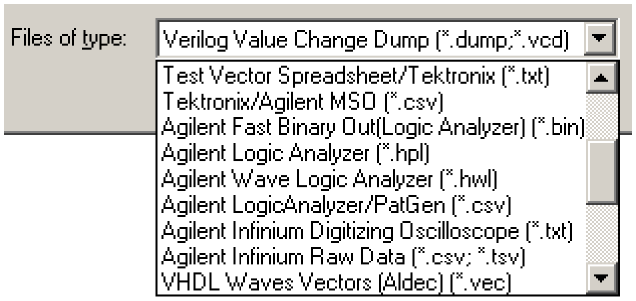
WaveFormer Pro can import waveform data from several simulators and pieces of test equipment. Usually, the *Import Waveforms* dialog opens and lets you manually choose which signals to load, however once a filter file is created and loaded it automatically determines which signals are loaded. Filter files are created by saving a diagram as a filter file. If this file is then set to be the "filtering file" for the program, any subsequently loaded diagrams will only load the signals contained in the filtering file. In addition, the signal attributes will be set using data from the filter file (e.g. the filter file can contain signal direction, radix, and pin mappings that will be applied to the signals from the files filtered by the filter file). Imports can also be done automatically using the batch mode feature discussed in Section 11.5 Batch Mode 159.

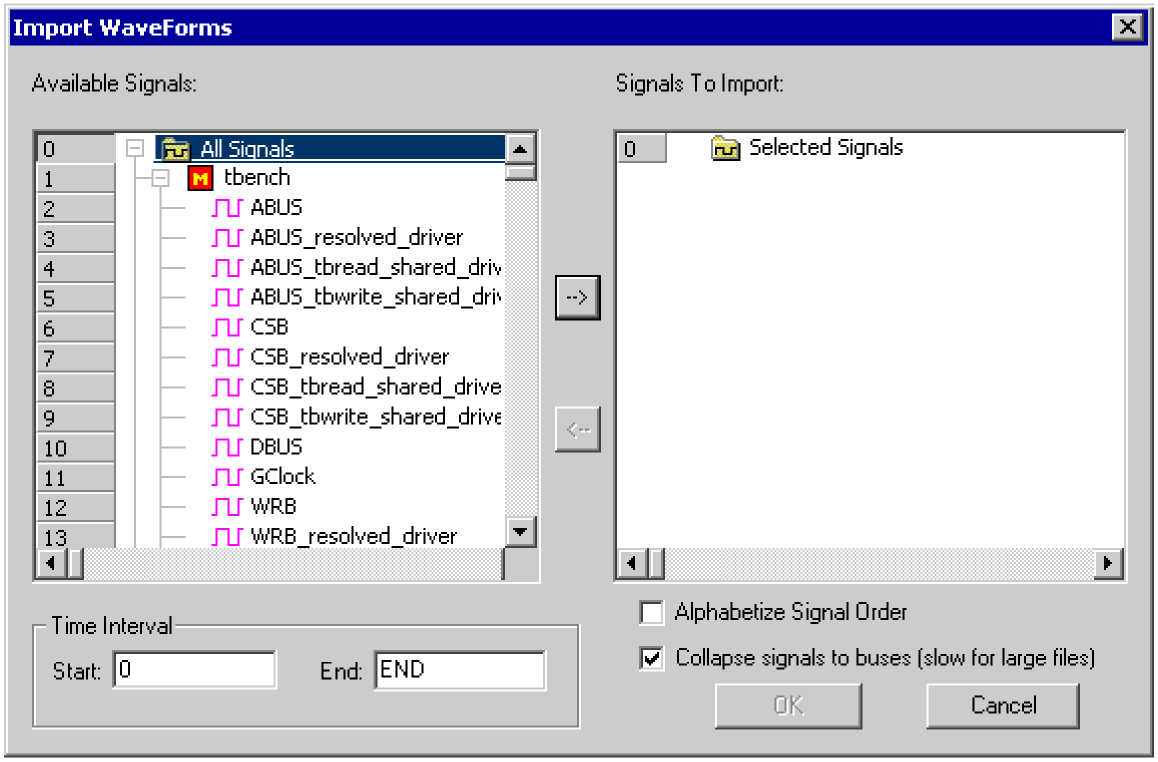
(Optional) Read Only Mode:

The read only mode may be used to ensure that changes to diagram waveforms can not be made when importing and exporting data. To turn on the read only mode, check the the **Options > Read-Only Mode** menu.

To import a file for the first time:

- Select the **Import/Export > Import Timing Diagram From** menu to open a special version of the *Open Timing Diagram* dialog that remembers the file type of the last file imported.
- Select an import format from the **Files of type** box, enter a file name and press the **Open** button, if no filter file is specified (see below) then the *Import Waveforms* dialog will open.


Files of type:
Verilog Value Change Dump (*.dump;*.vcd)
Test Vector Spreadsheet/Tektronix (*.txt)
Tektronix/Agilent MSO (*.csv)
Agilent Fast Binary Out(Logic Analyzer) (*.bin)
Agilent Logic Analyzer (*.hpl)
Agilent Wave Logic Analyzer (*.hwl)
Agilent LogicAnalyzer/PatGen (*.csv)
Agilent Infinium Digitizing Oscilloscope (*.txt)
Agilent Infinium Raw Data (*.csv; *.tsv)
VHDL Waves Vectors (Aldec) (*.vec)

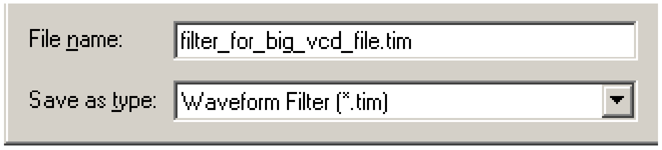


- Select the signals you wish to import, then press the **->** button to move them to the import list.

- If the **Alphabetize Signal Order** box is checked, then the signal names will be alphabetized when they are imported, otherwise they will be in the same order as stored in the file.
- If the **Collapse signals to buses** box is checked, then all numbered signals (e.g., bus0, bus1, bus2) to be collapsed to virtual buses (great for logic analyzer inputs).
- Also the time frame can be limited to a specific slice, if you enter values in to the **Start** and **End** boxes.

(optional) Create a Filter File after loading a waveform file:

- Save the loaded timing diagram by selecting the **File > Save Timing Diagram As...** menu, and selecting **Waveform Filter (*.tim)** from the *Save as type* box.



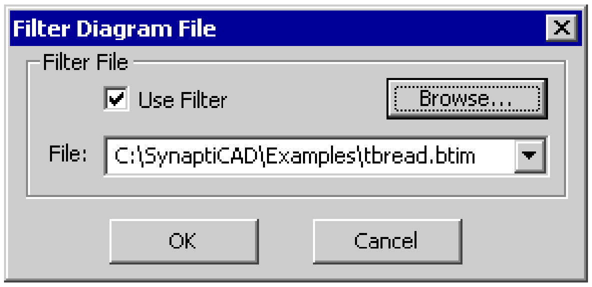
File name: filter_for_big_vcd_file.tim

Save as type: Waveform Filter (*.tim)

- You can also create a filter file by making a new timing diagram then copying and pasting signals into it.

Using the filter file:

- Choose the **Import/Export > Set Filter Diagram File** menu to open the *Filter Diagram File* dialog.
- Select the filter file and check the **Use Filter** box.



Filter Diagram File

Filter File

☑ Use Filter

File: C:\SynaptiCAD\Examples\tbread.btim

- While a filter file is set enabled, it will filter any waveform data that is loaded into WaveFormer, whether it is loaded by opening an existing timing diagram or by importing data from another file format. To disable the filter you only need to uncheck the **Use Filter** control.
- User-defined radices are also saved in the filter files, so you can store your own customized radix for viewing the imported waveform data from a VCD or other waveform source file.

Import File Formats

• VHDL and Verilog formats: The VCD is the standard output file format for all Verilog simulators and some VHDL simulators. The VCD parser automatically detects Extended VCD and properly handles the signal direction information. The other formats support specific VHDL simulators.	`Verilog Value Change Dump (VCD)`
	`SpeedWave VHDL`
	`Peak VHDL AWF`
	`VHDL Waves Vectors (Aldec)`

• Spice Formats are covered in [Section 8.6: Importing Spice waveforms](126) and include	`SPICE (csd and tr0)`

SPICE, HSPICE and OmegaSim for both binary and ASCII formats.	
• The TDML format is an industry standard XML format called the Timing Diagram Markup Language it is covered in Section 11.3 TDML Management 176.	`TDML`

• Third party simulators can be imported:
• Viewlogic's Viewsim simulator generates Workview WFM files.
• Capilano Computing's DesignWorks simulator generates DesignWorks files.
• Protel Technology's Advanced PLD simulator generates PLD files.
• Sysnopsys's TimeMill generates TimeMill OUT files.
	`Workview WFM`
	`DesignWorks`
	`Protel Advanced PLD`
	`Synopsys Time Mill`
	`Altera Timing Analyser Output`

• Static Timing Analysis formats:
• Altera Timing Analyser files can be read by this program
• Xilinx's place and route tool creates *.twx files which contain timing path delays through the FPGA. Timing parameter values can now be imported directly from these files so that you can analyze the timing paths through your post-layout design.
	`Altera Timing Analyser Output`
	`Xilinx Timing Analyser`
	`Xilinx Speed File`

• Test Equipment Formats:
• Agilent formats are covered in Section 11.6 Agilent Logic Analyzer - Import 165.
• Tektronix formats covered in Section 11.14 Tektronix Logix Analyzer Import 177.
	`Tektronix/Agilent MSO (csv)`
	`Agilent Fast Binary Out (Logic Analyzer)(bin)`
	`Agilent Logic Analyzer (hpl)`
	`Agilent Wave Logic Analyzer (hwl)`
	`Agilent Logic Analyzer/PatGen (CSV)`
	`Agilent Infinium Digitizing oscilloscope (txt)`
	`Agilent Infinium Raw Data (csv, tsv)`
	`Podalyzer Data`

• Spreadsheets can generate	`Test Vector SpreadSheet`

Copyright © 2011, SynaptiCAD Sales, Inc.

waveform data, see Section 11.11: Spreadsheets - Import and Export 173.	

Timing Diagram Formats:
• **Binary Timing Diagram** reads BTIM files that contain all signal and parameters as well as project options in a fast binary file format. This is the default file format for all timing diagrams.
	`Timing Diagram - Binary`
	`Timing Diagram`
	`Timing Designer`
	`Free Parm`

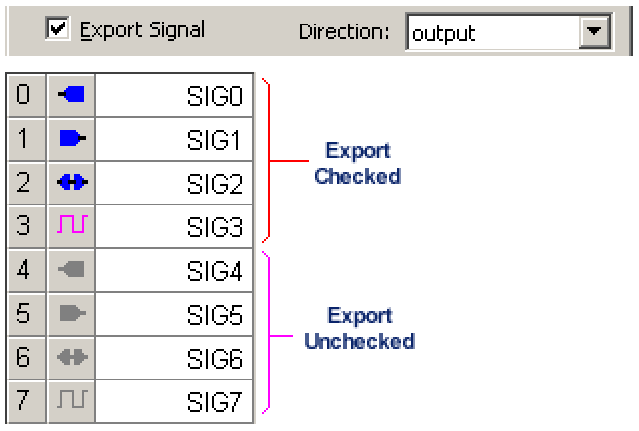
- **Timing Diagram** reads TIM files that contain all signals and parameters as well as project options settings in ASCII based format. This was the standard format used for all timing diagrams before version 8.0 of the SynaptiCAD product line. This format is still support but is slower loading than the btim format.
- **TimingDesigner** accepts TD timing diagram files from Chronology®'s timing diagram editor.
- **Text Free Parm** accepts txt files used for editing 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).
- Free Parm 138 reads FP files with just the free parameters information in Timing Project File Format. This is one way to make a library file.

11.3 Export General Instructions

WaveFormer Pro can generate stimulus files for many different simulators and third party tools. Exporting and file generation and also be done automatically using the batch mode feature discussed in Section 11.5 Batch Mode 159.

Prepare Signals for Export:

- By default, all signals will be exported. To exclude a signal from the export, double click on the signal name to open the *Signals Properties* dialog and uncheck the **Export Signal** box.
- Notice the colored direction icons show that the signal will be exported. The grey versions show that the export is unchecked and will not be exported.

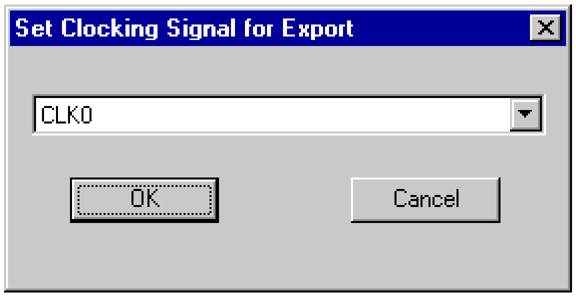


- If you are exporting to a pattern generator, then you may want to edit the pod mapping as described in Section 11.4 Map Signals to Test Equipment Pins 157.

If necessary, set the clocking signal for the export:

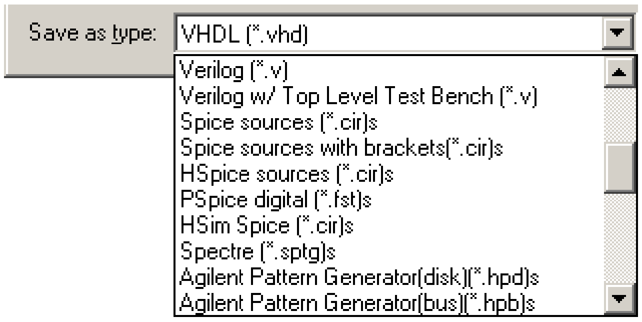
All of the pattern generator formats and some of the third party simulator formats require that the waveforms be sampled at regular intervals. The positive edge of the Export Clock Signal is used as the as the sampling edge, and by default this is the first clock in the timing diagram. However, any signal can be used as the Export Clock Signal by doing the following:

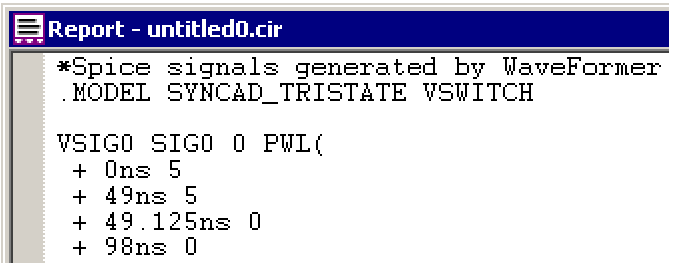
- Select the **Import/Export > Set Clocking Signal for Export** menu to open the *Set Clocking Signal for Export* dialog.
- Choose the signal that you wish to use for the clocking signal from the drop down list.
- Usage Trick: Putting grid lines on positive edges of the export clock helps show which signal values will be exported (See Section 2.3: Grid Lines on Clocks 44)



Export to a File:

- Select the **Import/Export > Export Timing Diagrams As** menu option to open a special version of the *Save As* dialog which remembers the file type of the last file exported.
- Select an export format from the **Save as type** box, enter a file name, and click **OK** to generate the file.
- **Note:** Once WaveFormer successfully creates the file it will display it in the Report window so that you can quickly verify that the file is correct. If you cannot see the Report window then choose the **Window > Report** menu to bring it to the front.


Save as type:
VHDL (*.vhd)
Verilog (*.v)
Verilog w/ Top Level Test Bench (*.v)
Spice sources (*.cir)s
Spice sources with brackets(*.cir)s
HSpice sources (*.cir)s
PSpice digital (*.fst)s
HSim Spice (*.cir)s
Spectre (*.sptg)s
Agilent Pattern Generator(disk)(*.hpd)s
Agilent Pattern Generator(bus)(*.hpb)s



```
*Spice signals generated by WaveFormer
.MODEL SYNCAD_TRISTATE VSWITCH

VSIG0 SIG0 0 PWL(
 + 0ns 5
 + 49ns 5
 + 49.125ns 0
 + 98ns 0
```

Export Formats:

• VHDL and Verilog Formats are covered in Section 4.4 Export VHDL and Verilog 65. These can create pure stimulus models or automatically wrap the testbench around the model. There is also some VHDL library features that effect the generated code.	VHDL VHDL w/ Top Level Test Bench Verilog Verilog w/ Top Level Test Bench
• Waveforms can also be exported directly to VCD to create files that can be read by other EDA tools	Verilog Change Dump (*.vcd)

and test equipment.	

• The Spice formats are covered in Section 8.5 Exporting SPICE Stimulus 124. Each signal has analog properties that determine how the export is accomplished. The Spice formats are generic formats and HSpice, PSpice, and HSim work with those programs.	`Spice sources`
	`Spice sources with brackets`
	`HSpice sources`
	`PSpice digital`
	`HSim Spice`

• The TDML format is an industry standard XML format called the Timing Diagram Markup Language it is covered in Section 11.13 TDML Management 176.	`TDML`

Third Party simulator and timing analysis formats are mostly straight ASCII output that are easy to figure out. Use the documentation from those tools and generate a few files to see how the code looks.
• The Altera format is clocked and is covered in Section 11.9 ALTERA Max Plus II Simulator - Export 170.
	`Model Force File`
	`ALTERA Vector format (Binary)`
	`ALTERA Vector format (Hex)`
	`Xilinx/Aldec/Orbit`
	`Workview CMD`
	`Mentor QuickSim II`
	`PLA Format`
	`Abel stimulus`
	`Abel Pins stimulus`
	`Spectre`

The Test Equipment Formats are all clocked file formats that require a clocking signal to be set in the diagram, and most also require a pod mapping to be defined. See:
• Section 11.7 Agilent Pattern Generator - Export 168
• Section 11.10 Pulse Instruments Pattern Generator - Export 171
• Section 11.12 STIL Test Vectors - Export 175
• Section 11.15 Tektronix Pattern Generator - Export 179
	`Agilent Pattern Generator(disk)`
	`Agilent Pattern Generator (bus)`
	`Agilent 167xx Pattern Generator (binary)`
	`Agilent 169xx Pattern Generator Full Mode`
	`Agilent 169xx Pattern Generator Half Mode`
	`PI-2005 Pattern Generator`
	`STIL Test Vectors`
	`Tektronix Test Vector Spreadsheet clocked`

Spreadsheet Formats are covered under the Section 11.11 Spreadsheets - Import and Export 173. These are all closely related formats and can be used for many different programs.	`Test Vector Spreadsheet`
	`Test Vector Spreadsheet without Timing`
	`Tektronix Test Vector Spreadsheet clocked`

SynaptiCAD Formats
• **Timing Diagram - Binary (*. btim)** is the default format for the program and it saves everything in a fast binary file format.
	`Timing Diagram - Binary`
	`Timing Diagram`
	`Free Parm`
	`Text Free Parm`
	`Text Free Parm with Margins`

• **Timing Diagram (*.tim)** is an older ASCII form of the the BTIM and was the standard format before version 8.0. These files can be manually edited.	`Timing Analysis Report`
	`Waveform Filter`

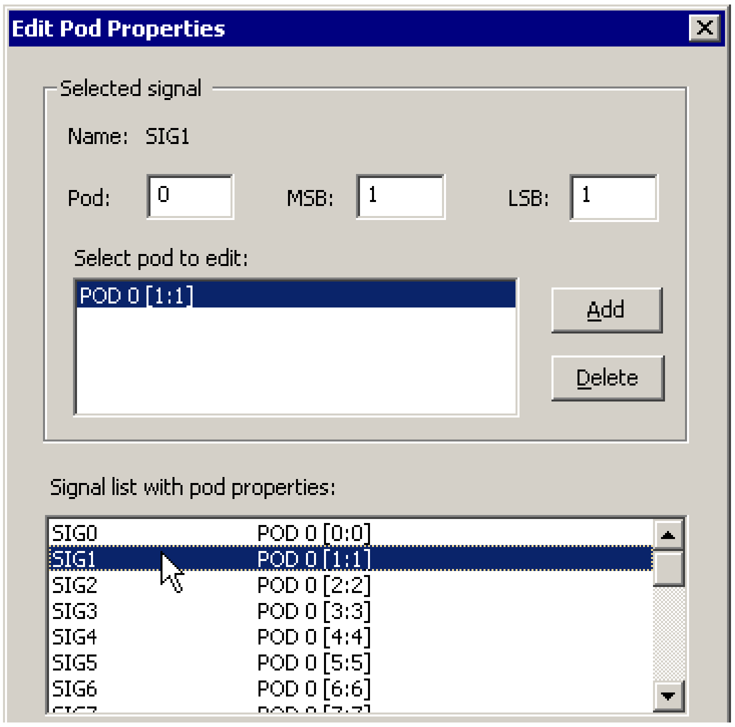
- **Free Parm (*.fp)** saves just the free parameters in the Timing Diagram format. This is one way to make a library file.
- **Text Free Parm (*.txt)** saves just the free parameters in a spreadsheet compatible format: comma or tab separated with a header (first line must be "NAME, MIN, MAX, COMMENT"). This is the standard way to make a library file. This is also compatible with Chronology® library files (formulas are not supported).
- **Text Free Parm with Margins** looks like the Text Free Parm format, but it includes an extra column with the margin information shown in the parameter window. This format is primarily for import into a spreadsheet for documentation purposes; it should not be used to create parameter libraries.
- **Timing Analysis Report** generates a report of all the timing calculations performed by the tool's timing analysis engine. It includes computations of parameter values, edge times, and setup and hold margin calculations. This report is for documentation purposes. See Section 1.5 Measuring Time and State values 26.
- **Waveform Filter** generates a "filter file" that contains only the signal properties, but now waveform data. Filter files can be used to selectively load waveform data from another waveform file and overwrite signal properties from that file. For example, a filter file can be used to load a subset of signals from a VCD file and set the input/output type of the signals. See Section 11.2 Import General Instructions 150 to show how to use filter files.

11.4 Map Signals to Test Equipment Pins

Normally signal information is exported in the order that it appears in the timing diagram. However, the *Pod Properties* dialog can define the signal order when exporting to the pattern generator formats. After the Pod Properties have been edited, they will be used instead of the default pod mapping. When the timing diagram is saved, the pod mapping will be saved as well. If any new signals are added to the timing diagram, they will be mapped in the default position. After the pod mapping has been set for a file, any signals that are subsequently deleted will be replaced with internal signals named $$FillerSignal#. The filler signals will be exported to preserve the pod mapping, but they will not be shown on the screen. Redoing the pod mapping will remove the filler signals.

To edit the Pod Properties of a timing diagram:

- Select the **Import/Export > Edit Pods** menu to open the *Edit Pod Properties* dialog. When the dialog comes up, it will have already mapped the signals to default positions.
- Click on a signal in the **Signal list with pod properties** list box. The signal name and bus size will be displayed in the upper portion of the dialog. The list of pods for that signal will be displayed in the *Select Pod to Edit* list box.


Edit Pod Properties

Selected signal

Name: SIG1

Pod: 0 MSB: 1 LSB: 1

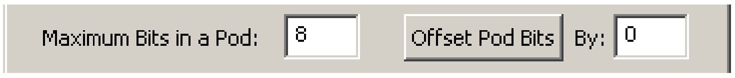
Select pod to edit:

POD 0 [1:1]

Signal list with pod properties:

```
SIG0    POD 0 [0:0]
SIG1    POD 0 [1:1]
SIG2    POD 0 [2:2]
SIG3    POD 0 [3:3]
SIG4    POD 0 [4:4]
SIG5    POD 0 [5:5]
SIG6    POD 0 [6:6]
```

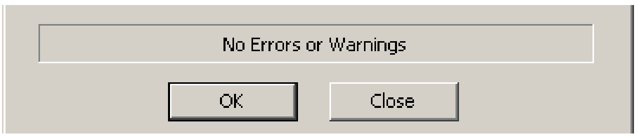
- To create a pod mapping for a signal, type in a new pod number in the **Pod** box, set the **MSB** and **LSB**, then press the Add button.
- The **Maximum Bits in a Pod** box specifies the width of the pod.


Maximum Bits in a Pod: 8 Offset Pod Bits By: 0

- The **Offset Pod Bits** button will shift the msb/lsb for all signals by the number specified in the **By** box.
- The **Clear Mappings** button clears all pod mappings.

- **To map pods to any empty signals:** First, select the signal you want to start to the automatic mapping from. Type in the pod, MSB and LSB that you wish to start the mapping at.

- Click the **Map Empty Signals** button.
- This will only map completely empty signals, so if a bus is partially mapped out and the *Map Empty Signals* button is pressed, the bus will not be changed.
- Pressing the **OK** button, makes the dialog check for errors. If errors are found the dialog will stay open and the error will be listed in the error box at the bottom of the dialog.


No Errors or Warnings

- Signal to pod mappings that contain

Pod Mappings Overlap means that there are two or more signals that are mapped to the errors will have an asterisk (*) displayed after their name in the *Signal list with pod properties* list box. The error message and asterisks will be displayed until the error has been resolved.

- If you do not want to fix the errors immediately, then press the **CLOSE** button and the dialog will close. However, as long as there are pod mapping errors the pattern generator export will not work.

same pod position.

Signal Mappings do not match signal size means that the pod mapping is either too large or too small for a signal.

Width is too large for current signal means that the pod you are trying to add is too big for the signal you are editing.

Pod number is too large means that the pod number entered in the Pod edit box is greater than the number of pods available to map signals to.

Bit Size is too large means that the bit number entered in either the MSB or the LSB edit box is greater than the size of the pods available.

Generate Tri-state Enable Signals to Turn Pods Off:

Some Pattern Generators support tri-state enable signals that allow an entire pod or group of signals to be alternately disabled and enabled during a test sequence. The tri-state enable signal is high when the associated pod is enabled, and low when the pod is tri-stated. Your particular pattern generator model will determine the rules involving the tri-state signals. To automatically generate a tri-state signal:

- Select the signals that you want to create tri-state enable signals for by clicking on the signal names.
- Choose the **Import/Export > Add Tri-State Enables for Selected Signals** menu option.
- Notice that a tri-state enable signal is created for each selected signal that has tri-stated data somewhere on the waveform.
- Use the pod-mapping features or arrange the signals so that the tri-state enable signals are exported to the correct pins for your particular pattern generator. Since the tri-state signals usually disable an entire group of pins, make sure that only the signals that you want to be disabled are mapped to those pins.

11.5 Batch Mode

All SynaptiCAD products have several command-line parameters which enable them to perform certain functions non-interactively. If you launch the product from a command window you can use one or more command line options to control how the product runs. You can also create start menu and desktop icon shortcuts that launch the tool with different options.

- To set the options passed when executing the tool from an icon or start menu, right click on the shortcut, select the Properties menu option, and add the desired options after the program name in the Target field of the shortcut.
- Batch mode diagram merge properties are set using the **Options > Diagram Merge Properties** menu (see Section 10.4 Merging Diagrams 144 for more information).
- Batch mode diagram compare properties are set using the **View > Compare Signals** menus (see Section 9.2 Viewing Compare Differences 131).

Most command-line parameters have both a short and a long form. For example, the long form of the option to control which product is launched is --product and the short form is **-p,** so the following are equivalent:

- syncad.exe --product wfp --input myfile.btim
- syncad.exe -p wfp -i myfile.btim

Below is a list of supported command line parameters. Each of these is described in detail later in this section including the valid values for string arguments.

Long Option	**Short Option**	**Argument Type**	**Description**
`--product`	`-p`	product code	Select which product to run
`--simulator`	`-S`	simulator code	Specify the default simulator
`--input`	`-i`	filename	Specify an input file containing waveform data.
none	*none*	filename	Specify a project file or timing diagram to load by listing the file name without any dash commands.
`--import`	`-m`	import code	Specify the file format of the file to load
`--output`	`-o`	filename	Specify the output file for batch conversion
`--export`	`-e`	export code	Specify the output file format for batch conversion
`--filter`	`-f`	filename	Specify a filter file to be used when loading files
`--eval-perl-file`	`-s`	filename	Invoke a perl script
`--eval-perl-string`	`-t`	string of perl commands	Invoke a perl command
`--batch-mode`	`-b`	none	Run without displaying the main window
`--quit`	`-q`	boolean	Quit after processing command-line parameters
`--corba-port`	`-c`	port-number	Connect to tcp port port-number on the localhost with the IOR
`--license`	`-n`	option name	specify which licenses to check out like ole or compare.
`-nosplash`		none	Runs the program without displaying the initial splash screen

`--report-crashes`	`-r`	never, ask	Asks whether to email a crash report before emailing SynaptiCAD

Write a Script if using more than one option:

If more than one command line option is used, we recommend that you put the commands in a perl script and call it using the **-s** *scriptname* option. See Appendix B: Writing Waveperl Scripts 191 for more information. A script will let you explicitly control the sequence of events. However, if more than one command line option is specified, they will be executed in this sequence:

1. Execute -f to setup any filter files.
2. Execute -i to setup any input files.
3. Execute -t to evaluate any perl commands.
4. Execute -s to execute a perl script file if specified.
5. If an output file (-o) and output file format(-e) are specified, the current timing diagram will be saved to that file in that format.
6. All other options

1) Select which product to run -p or -product

All SynaptiCAD products with a timing diagram graphical interface are distributed as one executable image (syncad.exe on Windows and syncad on UNIX). When you run syncad.exe, it will launch one of SynaptiCAD's products, depending on what command line arguments were present. For example, if the executable is invoked as **syncad.exe --product testbencherpro** it will load TestBencher Pro. The valid values for this option are:

Product	Valid Values
WaveFormer Lite	`wfl, wflite, waveformlite, or waveformerlite`
Timing Diagrammer Pro	`td, tdp, timing, or timingdiagrammerpro`
WaveFormer Pro	`wf, wfp, waveform, or waveformerpro`
VeriLogger Pro	`vl, vlp, vlog, vlogger, or veriloggerpro`
BugHunter Pro	`bh, bhp, bughunter, bughunterpro`
DataSheet Pro	`ds, dsp, datasheet, or datasheetpro`
TestBencher Pro	`tb, tbp, tbench, testbencher, or testbencherpro`

If no product option is specified on the command line, the program will start as whatever product was run most recently.

2) Set the default simulator -S or --simulator

Specify the default simulator used for new simulation projects. It takes one of the following arguments: verilogger_extreme, vhdl_extreme, vlogcmd, active_vhdl, active_verilog, modelsim_vhdl_se, modelsim_vhdl_nonse, modelsim_verilog, ncvhdl, ncverilog, vcs, vcsi, verilogxl, scirocco, mscl, or gcc.

3) Load an Input File -i or --input

The input command specifies an input file that contains waveform data.

- The filename can be either relative to the current directory or absolute.
- Can be used with the -m option which specifies the type of the file.

- Can be used with the -f option which creates a filter file that specifies which signals are loaded from the input file.
- Normally, this option will cause the file to load and display for interactive use, just as if you had started TestBencher Pro and then opened the file from the **File > Open Timing Diagram** menu. However, if you are doing a batch file conversion, using the -b command, the file will not be displayed.
- Alternative Approach: An input file can also be loaded using a perl function OpenFile( ) with the -s command.

4) Specify the format of an input file -m or --import

This option specifies the file format of the input file that is loaded with the -i option. If this is left unspecified, the program will guess at the file type by using the extension. This works for most files, but there are some formats like TXT which it cannot discriminate. This option is case-sensitive.

- To get a complete list of the valid values for this option, select the **Import/Export > Add/ Execute Script** menu to open a dialog, and then select the **Import** radio button. The valid values will be listed in the first column.
- As of this writing, the valid values are: AUTO_EXT_To_IF, TIM_To_IF, TDML_To_IF, FREEPARM_To_IF, FREETEXT_To_IF, WFM_To_IF, HPL_To_IF, POD_To_IF, ADPLD_To_IF, VCD_To_IF, HPWave_To_IF, AWF_To_IF, VHDL_To_IF, ALT_To_IF, GERMAN_To_IF, HPFASTBIN_To_IF, CSDF_To_IF, OSCOPE_To_IF, TECTRONIX_To_IF, TVECTOR_To_IF, chro.ipl, dworks.ipl, and acco.ipl.
- Alternative Approach: An input file can also be loaded using a perl function **twf::OpenFile( )** with the -s command. The optional second command of the function specifies the file import type.

5) Generate an Output file -o or --output

This option specifies the name of the output file when doing file format conversions.

- A -e command MUST also be used. This will specify the type of output file format to convert to.
- Alternative Approach: Use the perl function **twf::SaveFile( )** function with a -s command.

6) Output File Format -e or --export

This option sets the type of file to generate and save to the file listed with the -o command. If a -e is used, then there also MUST be a -o file specified.

- To get a complete list of the valid values for this option, select the **Import/Export > Add/ Execute Script** menu to open a dialog, and then select the **Export** radio button. The valid values will be listed in the first column.
- At the time of writing, the valid values are: IF_TO_TIM, IF_TO_FREEPARM, IF_TO_FREETEXT, IF_TO_FILTERTIM, IF_TO_TDML, wvhdl.epl, tvhdl.epl, verilog.epl, IF_TO_TDML, IF_TO_CMD, mentor.epl, spice.epl, hspice.epl, dspice.epl, IF_TO_CIR, orbit. epl, abel.epl, abelpin.epl, minc.epl, hpdisk.epl, hpbus.epl, HP_PATGENBIN, stil.epl, alterabin. epl, alterahex.epl, pla.epl, mtech.epl, tvector.epl, tvectornotiming.epl, tvectorclocked.epl, and logicexpress.epl
- Alternative approach: Use the perl function **twf::SaveFile( )** with a -s command.

7) Input File Filter -f or --filter

This filter file option specifies a filter file. See Section 11.2 Import General Instructions 150 for information on filter file creation and usage.

- Once the filter file is set, the same filter file will be used each time the product is run.
- To change or remove the filter file, use the -f option again. Specifying the empty string ("") indicates that no filter file should be used.
- This option does the same thing as the **Options > Set Filter File...** menu when running the program normally.
- Alternative Approach: Use the perl function **twf:SetFilterFile( )** function before the call to **twf:: OpenFile( )** inside the Perl script.

8) Evaluate Perl Script -s or --eval-perl-file

The **--eval-perl-file *filename*** option specifies a perl script file to be run. The script can be any valid perl script. The script will have access to all of the functions in product's perl interface. The perl script is evaluated in the main thread of the timing diagram editor so the application will be unresponsive while the script is running. All of the Perl commands are described in the file Twfapi.txt located in the **SynaptiCAD\Help** directory.

- The filename argument can be either an absolute path to the script or a path relative to the directory from which the timing diagram editor was run.

Below is an example of a perl script that could be called using the **-s** command line option. This code passes a filter file called **"filterfile.tim"** to the program, then loads a VCD waveform file called **"basename.vcd"**. The filter file lists the names of the signals to load from the VCD file. Next the script exports the waveform information to a spreadsheet format and saves the information in a file called basename.cbc.

```
#open a filter file that defines what signals to read from the input file
# as well as other information such as signal direction
twf::SetFilterFile("filterfile.tim");

#import the signals from a vcd file named "basename.vcd"
twf::OpenFile("basename.vcd");

#export the signals to a test vector spreadsheet file named "basename.cbc"
twf::SaveFile("basename.cbc","tvectorclocked.epl");
```

9) Evaluate Perl String -t or --eval-perl-string

The evaluate perl string function evaluates its argument as if it were the contents of a script. It can do everything that the -i, -e, and -m can do, plus much more. All of the perl commands are described in the file Twfapi.txt located in the **SynaptiCAD\Help** directory. Some examples are:

- The following command would launch TestBencher Pro and notepad at the same time (system is a perl function for launching a program, qw is a perl function that puts quotes around the string it encloses):

  syncad.exe --product testbencherpro --eval-perl-string "system(qw(notepad.exe))"

- The following command loads test.txt and specifies that it is a test vector spreadsheet file. It then loads test2.btim and performs a comparison between the files:

syncad.exe -t "twf::OpenFile(qw(test.txt),qw(TVECTOR_To_IF));twf::CompareFile(qw (test2.btim));"

- Another way to automatically compare these files is to embed the commands into a perl file:

syncad.exe --eval-perl-file temp.pl

if temp.pl contained

twf::OpenFile('C:\temp\test.txt', "TVECTOR_To_IF");

twf::CompareFile('C:\temp\test2.tim);

10) Enable Batch Mode -b or --batch-mode

If the -b option is specified, the program will not display the graphical interface. It will just run and perform the listed commands. This is useful when you want to invoke the SynaptiCAD product from within another program to convert a file from one format to another or do some other automated processing. This option interacts with the **--quit** option.

11) Exit After Processing Arguments -q

Setting this parameter's value to **true** causes TestBencher Pro to exit after it has carried out all the actions specified on the command line (anything triggered by the **--input,--output, --eval-perl-*,** etc). If it is set to **false**, TestBencher Pro will continue into interactive mode.

If you do not specify this argument on the command line, then it will determine what to do from the **--batch-mode** flag. If batch mode is enabled, it will default to **true**. If batch mode is not enabled, it will default to **false**. This is usually what you want, because if it is set to **false** in batch mode, TestBencher Pro will continue to run (invisibly) after doing all of the work you specified. (But then you *could* write a server in perl which would, say, accept requests on a socket, and you would want that to stick around after the initial processing was completed).

12) Set Corba port number -c or --corba-port:

Use mostly by OEM vendors, the -c option connects to tcp port port-number on the localhost with the IOR.

13) Pick License and options to run -n or --license

Normally the program will grab a license for the product and one of each available option packages (same as the "alloptions" code). When the -n option is used, only the options specified will be checked out.

- The following are valid options to specify: ole, compare, multidgm, gigawave, debuggertool, reactiveexport, trackertool, timinganalysis, timingparameters, alloptions or nooptions
- You must have the a valid license for the option in order to successfully run a program option.
- To ensure that the program checks out no optional features, specify "nooptions".

14) Run without the splash screen -nosplash

Runs the program but does not display the splash screen at start up.

15) Don't prompt when reporting crashes -r or --report-crashes:

The default behavior of the program is, -r ask, which cause a prompt to be displayed before emailing any crash information to SynaptiCAD. If -r never is set, then no crash report will be sent.

11.6 Agilent Logic Analyzer & Oscilloscope - Import

WaveFormer Pro can import data from the following Agilent logic analyzers and oscilloscopes. The following table shows the Agilent model type and the corresponding WaveFormer Pro file type.

Agilent Equipment	WaveFormer File Type
LogicWave	`Agilent Logic Analyzer (CSV)`

** 16900, 16800	`Agilent Fast Binary Out (Logic Analyzer)(bin)`
	`Agilent Logic Analyzer/PatGen (CSV)`

** HP16700/16600 Series Logic Analyzers (all logic analyzer cards including HP16550 & HP16555)	`Agilent Fast Binary Out (Logic Analyzer)(bin)`
	`Agilent Wave Logic Analyzer (hwl)`
	`Agilent Logic Analyzer (hpl)`

** HP 1660, 1670, 16500 ASCII	`Agilent Logic Analyzer (hpl)`

Infinium Digitizing oscilloscope	`Agilent Infinium Digitizing oscilloscope (txt)`
	`Agilent Infinium Raw Data (csv, tsv)`

Agilent MSO 6000/7000 (6012A) mixed signal oscilloscopes	`Agilent Mixed-signal oscilloscope (CSV)`

** See the below special instructions for getting the logic analyzer to create a WaveFormer compatible file.

HP16900 and LogicWave analyzers generate CSV files:

- Inside the logic analyzer save the data to a CSV file. For the HP16900 this is not the default file type so you have to choose CSV file.
- Inside WaveFormer Pro, choose **Import/Export > Import Timing Diagram From** menu to open the file dialog, and select **Agilent Logic Analyzer (*.csv)** from the files of type drop-down list box, select the file, and then click **OK** to open the file.

Use the HP16700/16600 Series Logic Analyzers (HP16550 & HP16555) to generate ASCII files

WaveFormer can import files ASCII and binary-based waveform files created by Aglient logic analyzers that run in the HP16700 and the HP16600 series timing analysis system. The default, waveform displays in the HP16555 and HP16550 generate files that have ASCII headers and binary waveform data that are NOT compatible with WaveFormer. However, there are two methods that use the Workspace and Listing windows to generate all-ASCII data. Either of the methods described in the next two sections can be used to generate WaveFormer Pro data files.

Method 1: Generate HWL files with the HP16700 Workspace window

The following instructions describe how to setup the Workspace window to generate ASCII and binary waveform files using the HP16700. The instructions may seem tedious, but once you have configured the options for a particular logic analyzer you can skip most of the beginning steps.

- Set up the logic analyzer and capture data. Click the **Workspace** button in the *Logic Analysis System* dialog. This will open the *Workspace* dialog. You should see a machine icon that represents the logic analyzer that captured the data.
- Drag the blue utilities icon called **File Out** to the working window and attach it to the machine that captured the data. You may have to expand the dialog to view the buttons at the bottom.
- Double-click on the **File Out** icon to open the *File Out* dialog.
- Enter a file name with an extension of **HWL** for ASCII or **BIN** for binary in the *File* edit box.

Note: Select a file name that is compatible with both the Aglient logic analyzer and the computer running WaveFormer Pro. File names of eight characters or less, no spaces, and no extra punctuation have the greatest compatibility.

- Next, select the type of file you are creating:

  For Binary:

  - Select the **Fast Binary** radio button.

  For ASCII:

  - Select the **ASCII** radio button and press the **ASCII Options** button to open the *ASCII Output Options* dialog.

- Determine the signal labels and the time range that you would like saved in the file. If you choose All as the time range than you may be generating very large files (greater then 50MB). Remember that most logic analyzers capture waveforms over a greater time range then is normally used by simulator test benches or timing diagram editors. Save only what you need.
- Click the **Close** button to close the *ASCII Output Options* dialog.
- Click the **Save Data** button, in the *File Out* dialog, to generate the file.
- Use the File Manager to verify that the file is the size that you expected it to be, then transfer the file to the computer running WaveFormer Pro.

Method 2: Generate HPL files with the HP16700 Listing Display window

An alternate method for generating files uses the Listing Display of the Aglient logic analyzer. The Listing Display allows the user to save data in different base formats, sample methods, and time ranges. The following instructions will help you create files that can be read by WaveFormer Pro. Use the Listing Display to generate ASCII waveform files:

- Set up the logic analyzer and capture data. Make a note of the sample time interval used to capture the data. You may need this later.
- Inside the logic analyzer *Machine* dialog, click on the **Navigate** button to open a series of pop-up menus. Choose the slot: **machine>listing menu tree** to open the *Listing* dialog
- Look at the data in the *Listing* dialog. Identify the trigger points and end points of the file.
- The middle column or columns contain the waveform data captured by the logic analyzer. The base readout controls the type of data that is displayed. WaveFormer Pro supports binary, octal, hex, decimal, and ASCII data types. To change the data type, Right-click on the base readout above the column.
- The last column is the sample time column. The sample column is displayed in either the Relative or Absolute time modes. To maintain timing accuracy, we recommend using the **Relative** time mode. In the Absolute time mode, the *Listing* window displays the times using a fixed number of digits. As time progresses, the display changes from nanoseconds to milliseconds and then to microseconds. This results in the truncation of decimal points. Two samples may appear to occur at 32us, when in reality one occurs at 32.1us and the other at 32.8us. By selecting the Relative time mode, you maintain data integrity.
- Verify that no column heads are selected. A selected column head has a yellow box around it. To deselect it, click on it. ***Warning:*** If a column is selected, only that column is saved to the file and WaveFormer Pro will not be able to read the file.
- Choose the **File > Print to File** menu option to open the *File* dialog.
- Enter a file name with an **HPL** extension in the file edit box. **Note:** Select a file name that is

compatible with both the Aglient logic analyzer and the computer running WaveFormer Pro. File names of eight characters or less, no spaces, and no extra punctuation have the greatest compatibility.

- Choose the time range that you would like saved to the file. If you choose All as the time range then you may be generating very large files (greater then 50MB). Remember that most logic analyzers capture waveforms over a greater time range than is normally used by simulator test benches or timing diagram editors. Save only what you need.
- Use the File Manager to verify that the file is the size that you expected it to be, then transfer it to the computer running WaveFormer Pro.

HP1660/1670/16500 Series Logic Analyzer generate HPL files:

WaveFormer Pro is capable of importing ASCII files generated by HP1660 and HP1670 series logic analyzers and from logic analyzers running on the HP16500 series frames. The following instructions were written specifically for the HP1670, however the HP1660 and HP16500 have similar interfaces. After capturing data with the logic analyzer, follow these steps to create the file:

1. Press the **List** button on the logic analyzer panel to see a listing of your data.
2. Look at the data in the *Listing* dialog. Identify the trigger points and end points of the file.
3. The middle column or columns contain the waveform data captured by the logic analyzer. The base readout controls the type of data that is displayed. WaveFormer Pro supports binary, octal, hex, decimal, and ASCII data types. To change the data type, Right-click on the base readout above the column.
4. The last column is the sample time column. The sample column is displayed in either the Relative or Absolute time modes. To maintain timing accuracy, we recommend using the **Relative** time mode. In the Absolute time mode, the *Listing* window displays the times using a fixed number of digits. As time progresses, the display changes from nanoseconds to milliseconds and then to microseconds. This results in the truncation of decimal points. Two samples may appear to occur at 32us, when in reality one occurs at 32.1us and the other at 32.8us. By selecting the Relative time mode, you maintain data integrity.
5. Click on **Print** at the top of the screen to open the print menu.
6. Select **Print Disk** from the list of menu options to open the *Print to Disk* dialog.
7. Enter a file name with an **HPL** extension.
8. Click the **Output Format** button and select either **ASCII (ALL)** or **ASCII (PARTIAL)** depending on how much data you want to save. If you select a time range of ALL, the files you generate may be very large (greater then 50MB). Remember that most logic analyzers capture waveforms over a greater time range than is normally used by timing diagram editors or test bench simulators. Save only what you need.
9. If you selected ASCII (PARTIAL), fill in the **Start State** and **End State**. These refer to the sequential listing of numbers on the left-hand side of the data listing.
10. Click the **Output Disk** button and select either Hard Disk or Flexible Disk. If you choose the hard disk and later want to copy the file to a flexible disk, follow the instructions in Section 2.
11. Click the **Execute** button to save the file.
12. In WaveFormer, use the **Agilent Logic Analyzer (hpl)** import type when importing

11.7 Agilent Pattern Generator - Export and Import

WaveFormer Pro can create stimulus files that can be imported into Agilent Pattern Generators. It can also import CSV files from 16822A and 16900 series. The following equipment is supported. Please call for information on the latest Agilent test equipment.

Agilent Equipment	WaveFormer File Type

16900 Series Timing Analysis System	`Export with:`
	`  Agilent 169xx Pattern Generator Full Mode`
	`  Agilent 169xx Pattern Generator Half Mode`
	`Import into WaveFormer using:`
	`  Agilent Logic Analyzer (CSV)`

16822A pattern generator	`Export with:`
	`  Agilent 169xx Pattern Generator Full Mode`
	`  Agilent 169xx Pattern Generator Half Mode`
	`Import into WaveFormer using:`
	`  Agilent Logic Analyzer (CSV)`

16700 Series Timing Analysis System with 16720 A pattern generator	`Agilent 167xx Pattern Generator (binary) (PGB)`

HP 16522A pattern generator card runinnd on the 16500 or 16600 Series Timing Analysis Systems	`Agilent Pattern Generator(disk)`
	`Agilent Pattern Generator (bus)`

The most current and detailed instructions can be found on-line at ttp://www.syncad.com/hpdetail.htm.

WaveFormer Instructions (VERY IMPORTANT)

The following are rules and tips for creating timing diagrams that export to the different pattern generator formats.

- **Set the clocking signal:** Make sure the timing diagram contains one clock signal, which will be used as the sampling clock. At each rising edge of the clock, the states of each signal will be written out to the clocks will be ignored. For more information see 11.3 Export General Instructions 154 and the subsection on clocking signals.
- **Must have 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).
- **Map the signals to the pattern generator pins** using the *pod mapping dialog* described in section 11.4 Map Signals to Test Equipment Pins 157.
- **Drawing the waveforms:** 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.
- **Setup Tri-state enable signals:** You can include tri-state enable signals in the exported data. To do this, select the signals that you want to create tri-state enable signals for and select the **Import/Export > Add Tri-State Enables for Selected Signals** menu option. This

function will scan through the data signals and find each segment that is tri-stated or un-driven (blue) and create a high segment on the tri-state enable signals to match those segments. See 11.4 Map Signals to Test Equipment Pins 157.

- The end of the timing diagram is determined by the last edge of the longest drawn signal or an **End Diagram Marker** (note 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.

HP16900 Notes

WaveFormer generates files that are directly compatible with the 16900. However, the Agilent 16900 will accept 16700 PGB binary files that have been translated using an Agilent program that ships with you logic analyzer. The PGB Translator program located in your logic analyzer software at **All Programs > Agilent Logic Analyzer > Utilities >Pattern Generator > 167xx Pattern Generator PGB Translator.**

HP16522 Notes

- The HP16522A pattern generator has a limit of 126 signals per stimulus file, and a Group Bus limit of 32 signals per bus.
- 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.
  - The ***.hpd** format is used for file transfer via network or diskette.
  - The ***.hpb** format uses the HPIB bus for file transfer.

11.8 Agilent and HP file transfer instructions

How you transfer files to and from the your HP system will depend on the system and how your network is configured. The following sections describe methods that have worked for us, and may not represent the most efficient transfer methods for your system.

HP16700 timing analysis system to another computer

We have used two different methods to transfer files from the HP16700 timing analysis system.

Using a diskette:

- From the window titled *16700A Logic Analysis System*, click on the button labeled **File Manager** to open the file open dialog.
- Navigate to the directory your file is in and then select it by clicking on it.
- Select the **File > Copy** menu option to open a dialog that prompts you for the destination.
- Click on the button to the right of Current Disk to select the **Flexible Disk** option. Fill in the remaining information and click **OK**.

Using the network:

- From the window titled **16700A Logic Analysis System**, click on the button labeled **System Administration** to open the *System Administration Tools* dialog.
- Click on the button labeled **FTP** to open the IP Address message box.
- Type in the IP address of the system you want to transfer the files to. This system must have an FTP server running on it.

- Click **OK** to close the message box and open the *Terminal* dialog.
- From the HP 16700 terminal window, log in to your ftp site (server) as you normally would.
- Use the standard commands **put *filename*** to transfer a file to the site, or get ***filename*** to transfer a file from the site to your HP unit.

HP1660/1670 series logic analyzers to another computer

The HP1660/1670 logic analyzers can transfer files either by using a diskette or by using the network connection. Use the following procedure to transfer files using a floppy disk:

- Press the **System** button on the logic analyzer. This brings up a window showing how your logic analyzer is connected to the outside.
- Click on the **External I/O button** and select **Hard Disk**. You will get a listing of the files on the hard disk.
- To copy a file from the hard disk to a floppy disk: edit the blue dialog boxes to resemble the figure below, then click the **Execute** button to copy the file.

11.9 ALTERA Max Plus II Simulator - Export

ALTERA Vector Format (*.vec) saves waveforms to a format used by the Altera Max PlusII simulator. It is a clocked format.

Rules for the Atera Vector Format Files:

- 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.
- You may want to select the signal that you want to be the clocking signal (See 11.3 Export General Instructions: Set Clocking Signals 154).
- 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.

The following table describes how 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;

To use an altera vec file:

1) Inside Max PlusII, select the **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, and press the **OK** button to close the *Inputs Outputs* dialog.

4) 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.

5) **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.

11.10 Pulse Instruments Pattern Generator - Export

WaveFormer Pro can export waveform data into a format that can be read by Pulse Instruments' pattern generator PI-2005. WaveFormer exports the waveforms as displayed in the timing diagram window. TestBencher supports repeat loop markers that will cause a section of the waveform data to be repeated by the pattern generator.

Each timing diagram should contain one clock. The positive edges of the clock will be used as the sampling points for rest of the waveforms. The period of the clock will be output to the **PERIOD** statement in the PI-2005 file. By default, WaveFormer will use the first clock in the timing diagram as the sampling signal. If you import waveforms from a simulation file you will need to either add a clock or convert the clocking signal to a SynaptiCAD clock by right clicking on the signal name and choosing **Signal <=> Clock** from the context menu.

The PI-2005 requires that waveform values be only 0's and 1's. So the export script converts non-binary waveform values like tri-state into 1's during the export. Also the PI-2005 only supports time units of **NS**, **US**, and **MS** for the **PERIOD** statement. This means that the display time unit for the timing diagram should be set to one these three values.

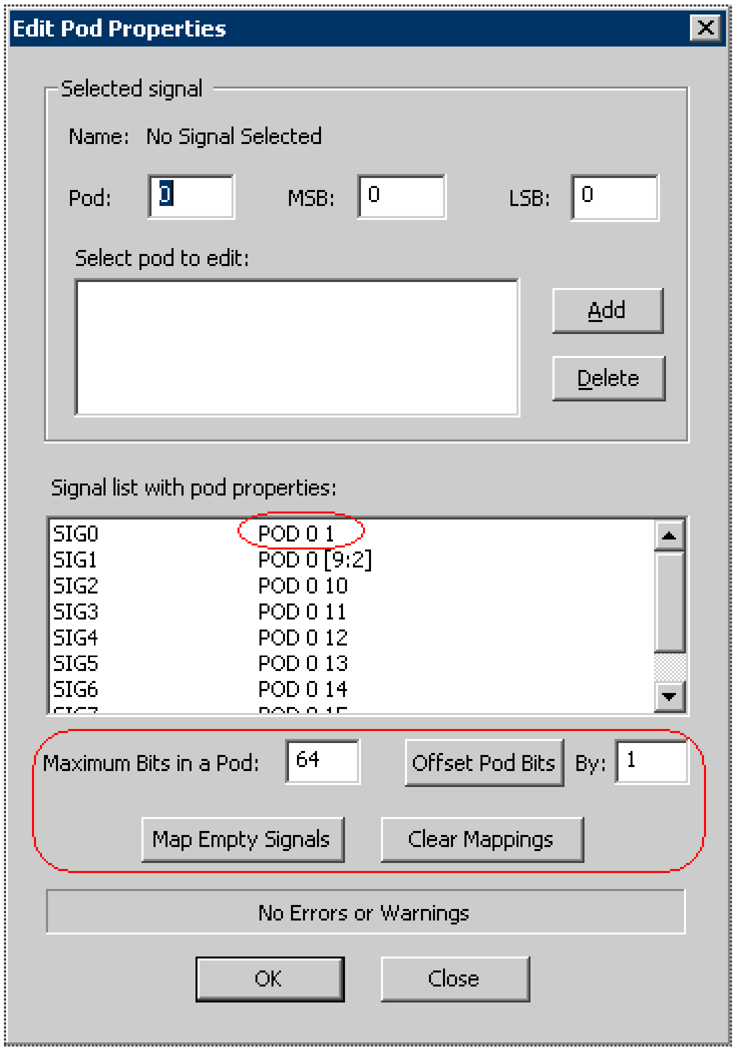
Data is loaded serially one channel at a time into the PI-2005. The signals in the timing diagram are mapped to the channels in the PI-2005 using the Edit Pod Properties dialog. PI-2005 channels start with #1 and use Pod0. You must setup Pod0 so that the maximum pod bits is 64 and the offset is 1.

TestBencher supports repeat-loop markers in the export of PI-2005 data (See Section 5.5: Loop Markers in the *TestBencher Pro Manual*). Each repeat loop defines a subpattern in PI-2005 file. The smallest subpattern supported by the instrument is 20 clock units long, so each repeat loop should span at least 20 clock pulses. Repeat loops are ignored by WaveFormer Pro.

WaveFormer Pro Instructions:

- Either load or create a timing diagram in the timing diagram editor.
- Make sure the timing diagram contains one clock signal, which will be used as the sampling clock.

- Choose **Import/Export > Edit Pods** menu to open the *Edit Pods* dialog.
- Set the **maximum pod bits** to **64** and the **offset** to 1.
- Press the three buttons in this order, **Clear Mappings**, **Map Empty Signals**, and **Offset Pod Bits**. Check to make sure that the first signal mapping is on POD0 and Channel 1. All the other signals should be mapped to POD0.
- Press **OK** to close the dialog.



Name: No Signal Selected

Pod: 0 MSB: 0 LSB: 0

SIG0	POD 0 1
SIG1	POD 0 [9:2]
SIG2	POD 0 10
SIG3	POD 0 11
SIG4	POD 0 12
SIG5	POD 0 13
SIG6	POD 0 14

Maximum Bits in a Pod: 64

Offset Pod Bits By: 1

No Errors or Warnings

- Select the **Import/Export > Export Timing Diagram As** menu option to open a file dialog.
- Choose **PI-2005 Pattern Generator** from the file type drop-down list box.
- Type a file name and click the **OK** button to close the dialog.
- The file will be displayed in the *Report* window. Also any errors or warnings found during export will be displayed in the **waveperl.log tab** in the Report window.

PI-2005 Pattern Generator Instructions

The resulting text file is a series of GPIB commands that can be read directly by the PI-2005 pattern generator. Contact Pulse Instruments, support@pulseinstruments.com, for details on custom application development and working directly with the instrument. The exported text file can also be imported into the PI-PAT control software.

PI-PAT is the control software and pattern editor for the PI-2005:

1. In the PI-PAT software, create a new PI-2005 document via **File > New**.
2. Choose **File > Import File** menu and select the exported text file.
3. Once the pattern file has been imported, PI-PAT may then be used to control the pattern

generator and to edit the waveform patterns.

11.11 Spreadsheets - Import and Export

The Spreadsheet formats are compatible with spreadsheet programs(like Excel), because the waveform data is stored in a row and column format with the elements in the rows being separated by TAB characters. This format our most generic format and has been adopted by many other companies and products, the most notable is that it is also the format for Tektronix logic analyzers and pattern generators. Also most mathematical analysis products can produce files that are in this format.

Importing and Exporting

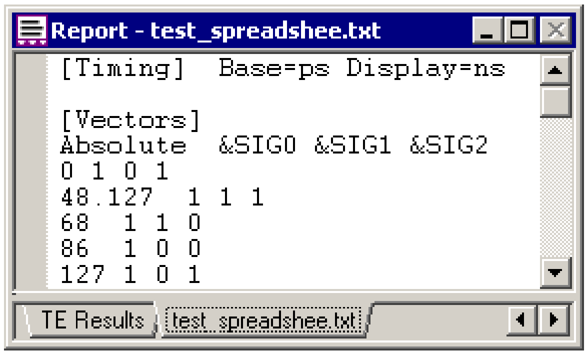
- To Import, select the **Import/Export > Import Timing Diagram From** menu and choose **Test Vector SpreadSheet** type from the type box. WaveFormer will determine the type of file from the information in the header.

```
Test Vector SpreadSheet
```

- To export, select the **Import/Export > Export Timing Diagrams As** menu and pick one of the test vector spreadsheet types

```
Test Vector Spreadsheet
Test Vector Spreadsheet without Timing
Tektronix Test Vector Spreadsheet clocked
```

- The **test vector spreadsheet** outputs a tab-separated row for each change on any signal. The first column shows the time of the change, and there is is one column per signal.
- The **test vector spreadsheet without timing outputs** outputs the same format, except that it does not include the time column.



```
[Timing]  Base=ps Display=ns

[Vectors]
Absolute  &SIG0 &SIG1 &SIG2
0 1 0 1
48.127  1 1 1
68  1 1 0
86  1 0 0
127 1 0 1
```

- The **Tektronix test vector spreadsheet clocked** samples the states of the signals on the export clocking signal edge, so it doesn't create a row for each change on any waveform, instead it creates a row for each positive clock edge.

General instructions for generating a file from a spreadsheet program:

- Inside the spreadsheet program, save your test vector file as a **tab delimited table**. Sometimes the default setting maybe comma or some other character, so look for a tab type of format.

Test Vector Spreadsheet Format

The Test Vector Spreadsheet format 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 which is ignored by the tool, followed by any number of sections and comments. The header can be arbitrarily formatted, since WaveFormer ignores any text until it encounters a section heading.

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 and its contents will be ignored. Note that sections and signal names are case sensitive.

[Timing] Base=*time_base* Display=*time_base*

- The base time is the smallest time that is represented by the file. And the Display is the units that time values are written in. The base unit should be at least an order of magnitude smaller than the display unit. Valid units are **fs**, **ps**, **ns**, **us**, and **ms** (default is "Base=ps" , "Display=ns").

[Clocks]

- The first line after the section header lists the column 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).

[Clocks]				
Name	Period	Offset	Duty	Invert
CLK0	100	0	50	0

[Vectors] Radix=*type_of_radix* End=*some_integer*

- **Radix=*hex*** provides a default radix for the file. Should be a valid radix abbreviation (eg, hex, bin, dec)
- **End=50** extends signals this far beyond the last event. Defaults to 50 past the last event if not used.
- **Title Line:** The line following the section header must be a title line consisting of **Absolute**, **Relative**, ***all of the signal names*** to be used in this table, and ended with **Comment**, each item in its own cell (or separated by TAB characters). Note that a signal probably should not be in two different vector sections since this will cause two signals with the same name to appear in WaveFormer.
- **Signal Names** are defined in the form **[@ | &]signalname*[msb:lsb]***. Each signal name starts with an optional special character that determines its direction-in when output to certain formats. The '@' in front causes the signal to be an input, whereas '&' 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 [msb:lsb] form (i.e. [7:0] is an eight bit vector). If no size is given, it is assumed to be a single bit vector. Neither of these 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.

[Comment]

- Declares a comment block. Anything inside this section is ignored.

[End]

- Ends WaveFormer input. When read by WaveFormer Pro, DataSheet Pro or TestBencher Pro, this section header is equivalent to the end of the file. No section beyond this point will be read.

Example Spreadsheet:

```
This text is in the Header section. 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.

 [Vectors]   Radix=dec      End=50

 Absolute    Relative       &SIG1[0:7]    @SIG2  &SIG3     @SIG4      Comment

 0           0              1             0      x         1          Start the 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

[Comment]
This section is ignored. Anything can be put in this section.

 [Clocks]

 Name        Period     Offset        Duty      Invert

 CLK1        20         5             10        0

 CLK2        25         10            15        1

[End]   Nothing after this line is read into WaveFormer
```

11.12 STIL Test Vectors - Export

STIL format is compatible with the IEEE Standard P1450 Standard Test Interface Language and it is used to produce files for test equipment. WaveFormer Pro can generate stimulus vector files in this format. Below is a list of rules and objects that generate code:

- This is a clocked format so the timing diagram must have at least one clock and one or more signals with waveforms. See 11.3 Export General Instructions 154 and the subsection on clocking signals. 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 testbench (inputs come from the device under test, or tester, and outputs are generated by the testbench 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' (Waveform: high state),

  '0' => 'ForceDown' (Waveform: low state),

  'N' => 'ForceUnknown' (Waveform: valid or invalid state),

  'P' => 'ForcePrior' (Waveform: unknown state),

  'Z' => 'ForceOff'(Waveform: tri state),

  Output Signals (blue) :

  'H' => 'CompareHigh' (Waveform: high state),

  'L' => 'CompareLow' (Waveform: low state),

  'X' => 'CompareUnknown' (Waveform: valid, invalid or unknown state),

  'T' => 'CompareZ' (Waveform: tri state),

  InOutput signals (black and blue) : Input and Output signals representation.

11.13 TDML Support - Import and Export

The Timing Diagram Markup Language, or TDML, an industry standard xml format for describing timing diagrams and timing parameters. It is recommended by the SI2 ECIX committee and SynaptiCAD has been working with Si2 on this format since November of 1996. Keep checking our web site for updates on the newest TDML information. SynaptiCAD's tools can read TDML files produced by other EDA tools and also write information out to a TDML format. Note however, when working within the SynaptiCAD tool suite the perferred format is BTIM which covers more advanced features like simulated signals and more graphical objects then can be described using TDML.

To import a TDML timing diagram:

- Select the **Import/Export > Import Timing Diagram From** menu to open a special version of the *Open Timing Diagram* dialog that remembers the file type of the last file imported.
- Choose **TDML File (*tdml,*tdm)** format from the **Files of type** box, enter a file name and press the **Open** button to load the file.

To export a TDML timing diagram:

- Select the **Import/Export > Set Clocking Signal for Export** menu to open the *Set Clocking Signal for Export* 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.
- Choose **OK** to save the file. The file will also be visible in the *Report* window.

11.14 Tektronix Logic Analyzer & Oscilloscope - Import

WaveFormer Pro can read ASCII files created by the TLA 700 series logic analyzer and by the MSO4000 digitizing mixed signal oscilloscope series.

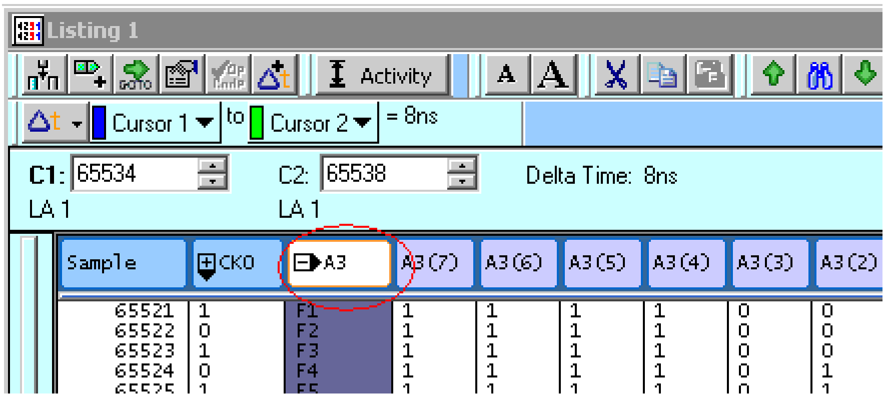
Digitizing Mixed Signal Oscilloscope Instructions

WaveFormer Pro can read files generated by the MSO4000, MSO3000, and MSO2000 series oscilloscopes.

- Using the Tektronix oscilloscope save the data to a CSV file.
- Inside WaveFormer Pro, choose the **Import/Export > Import Timing Diagram From** menu to open a special version of the *Open Timing Diagram* dialog that remembers the file type of the last file imported.
- In the **File of Type** box, choose the **Tektronix/Agilent MSO (*.csv)** file type.
- Then open the file you created with the oscilloscope.

TLA 700 Instructions

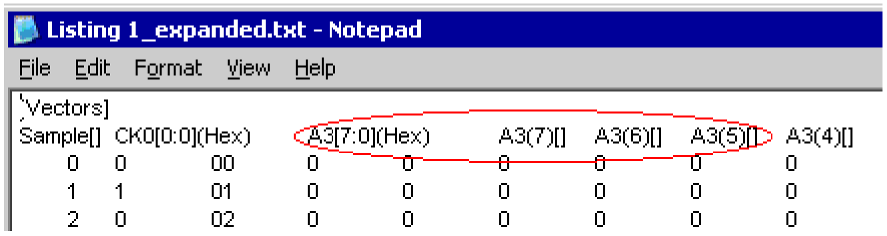
- Set up the logic analyzer and capture some data.
- Display the data in the **Listing Window**. Note: the TLA 700 Waveform window can only export bitmap pictures of the waveforms so it cannot be used to generate a WaveFormer Pro compatible file.
- Set up the timestamp column to display relative (previous) times. To do this, Right-click on the **Timestamp** column and choose **Properties** from the context menu. Then choose the **Column** tab in the *Properties* dialog. In the *Timestamp Reference* area choose **Previous** from the drop-down list box.
- Set up the Listing window so that there is one *Timestamp* and columns containing groups with data that should be exported. The data columns should be formatted in either hex, binary, or decimal form, and the timestamp in previous/relative mode. To remove excess columns, like the **Sample** column, Right-click on the column and then choose **Delete Column** from the context menu.
- In the newest version of the TLA software, the listing window exports what you actually see in the window. That means that only groups names will be exported, unless you press the the + button to expand the group. In the expanded state, both the group and the channel names will be exported. Inside WaveFormer Pro, you may wish to delete the group signal because it is a duplicate of the information contained in the channels.



Cursor 1 to Cursor 2 = 8ns

C1: 65534 LA 1 C2: 65538 LA 1 Delta Time: 8ns

Sample	CK0	A3	A3(7)	A3(6)	A3(5)	A3(4)	A3(3)	A3(2)
65521	1	F1	1	1	1	1	0	0
65522	0	F2	1	1	1	1	0	0
65523	1	F3	1	1	1	1	0	0
65524	0	F4	1	1	1	1	0	1



```
[Vectors]
Sample[] CK0[0:0](Hex)   A3[7:0](Hex)   A3(7)[]  A3(6)[]  A3(5)[]  A3(4)[]
   0     0      00       0       0        0        0        0        0
   1     1      01       0       0        0        0        0        0
   2     0      02       0       0        0        0        0        0
```

- Select the **File > Export Data** menu option to open the *Export Data* dialog.
- Choose **Tektronix TLA Data Exchange Format (*.txt)** from the *Save as Type* list box.
  - If the **Tektronix TLA Data Exchange Format** is missing, then you have an older version of the TLA software. The file can be manually generated by (1) choosing **Text File (*.txt)**, then (2) pushing the **Options** button to open the *Export Data Options* dialog. Next, check the three checkboxes labeled: **"Include Column Headers"**, **"Use Enhanced Headers"**, **"Include Unit Characters"**. Check the **Tab** radio button in the *Field Delimiter* area and push **OK** to close this dialog.
- Enter a ***file name*** and push the **Save** button to save the file.
- Export file to the machine containing WaveFormer Pro. Note: if you are running WaveFormer on a UNIX machine you must ensure the file is correctly moved using a utility that will preserve the line feeds.

WaveFormer Instructions

- In WaveFormer Pro, select the **Import/Export > Import Timing Diagram From** menu option to open the file dialog, and choose **Test Vector Spreadsheet** file from the *file type* drop-down list box.

What a WaveFormer-Ready Tektronix file looks like:

The default settings of the Tektronix Logic analyzer do not always produce a file that can be read by WaveFormer Pro. If you are having trouble reading the file, load it into an editor and look at the file. Here is a quick list of things to look for:

- Only one header line.
- Either the (radix) information is included in header or the all the data is hex values.
- Timestamp column is present and the time appears to be in relative or previous mode. Absolute time is not supported.

Below is a sample of a valid file.

```
[Vectors]
 Hex[3:0](Hex)  Decimal[3:0](Decimal)  Binary[3:0](Binary)  Timestamp[ ]
 C              12                     1100                 0 ps
 D              13                     1101                 10.000 ns
 E              14                     1110                 10.000 ns
 F              15                     1111                 10.000 ns
 0              0                      0000                 10.000 ns
 1              1                      0001                 10.000 ns
 2              2                      0010                 10.000 ns
 3              3                      0011                 10.000 ns
 4              4                      0100                 10.000 ns
 5              5                      0101                 10.000 ns
 6              6                      0110                 10.000 ns
 7              7                      0111                 10.000 ns
 8              8                      1000                 10.000 ns
 9              9                      1001                 10.000 ns
 A              10                     1010                 10.000 ns
 B              11                     1011                 10.000 ns
```

11.15 Tektronix Pattern Generator - Export

WaveFormer Pro can export waveform data into a format that can be read by the Tektronix pattern generator TLA7PG2 and the Tektronix DGLink software. DGLink takes the WaveFormer file and converts it into a binary file that can drive the DG2020A, DG2030, and DG2040 data generators.

WaveFormer Instructions

- Either load or create a timing diagram in WaveFormer.
- **Set the clocking signal:** Make sure the timing diagram contains one clock signal, which will be used as the sampling clock. At each rising edge of the clock, the states of each signal will be written out to the clocks will be ignored. For more information see 11.3 Export General Instructions 154 and the subsection on clocking signals.
- **Optional Tri-State Enable Signals:** You can include tri-state enable signals in the exported data. To do this, select the signals that you want to create tri-state enable signals for and select the **Import/Export > Add Tri-State Enables for Selected Signals** menu option. This function will scan through the data signals and find each segment that is tri-stated or un-driven (blue) and create a high segment on the tri-state enable signals to match those segments.
- Select the **Import/Export > Export Timing Diagram As** menu option to open the save as dialog, and choose **Tektronix Test Vector Spreadsheet Clocked** from the *file type* drop-down list box.
- Type a file name and click the **OK** button to close the dialog. This will generate the file and also display it in the *Report* window. If you cannot see the Report window choose the **Window > Report** menu option to bring it to the front.

Tektronix Pattern Generator Instructions

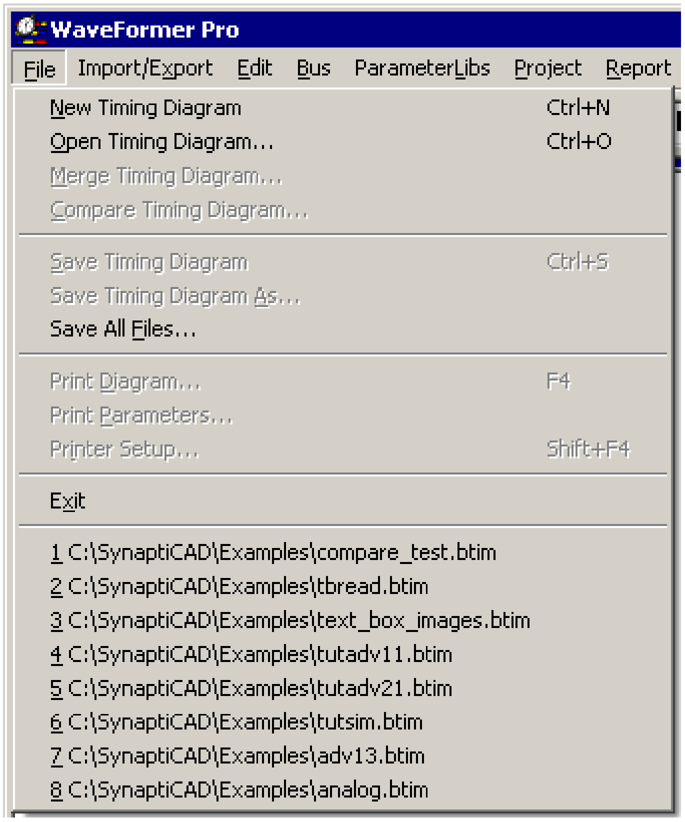
- In the *Systems* window, push the **Prog** button on the Pattern Generator to open *Program* dialog.
- Click the **Block** tab, and select the block number to edit.
- Click the **Edit Pattern** button to open the *Block Listing* window.
- Select the **File > Import Data** menu option to open the file dialog.
- Choose **TLA Text File** from the *file type* drop-down list box.
- Browse and find the file that you saved from WaveFormer Pro.
- Type a file name and click the **OK** button to close the dialog.

Appendix A: Menus and Colors

The following is a quick overview of the different menu functions and the colors in the program.

File Menu

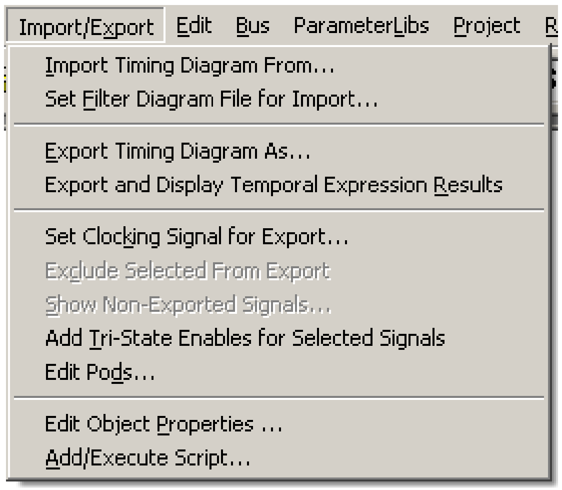
- **New Timing Diagram** opens a new timing diagram and assigns it a default name.
- **Open Timing Diagram** opens an existing timing diagram.
- **Merge Timing Diagram** copies the contents of the specified and adds them to the current timing diagram. Duplicate signal and normal parameter names are changed. Duplicate free parameters overwrite the existing free parameters, so always make corrections to the *free parameter library* files before loading into a project. See Section 1.7 Copy Signals and Waveforms 31 and Section 10.5 Self-Contained Timing Diagrams 146.
- **Compare Timing Diagram** copies the signals of the specified file, sets the compare flag on the signals, and then adds them to the current timing diagram. It also causes a comparison to be run. See Chapter 9: Waveform Comparisons 129.
- **Save Timing Diagram** saves the current timing diagram. By default this always saves to a BTIM format even though other format can be saved from this file dialog. See Section 11.3 Export General Instructions 154.
- **Save Timing Diagram as** saves the current timing diagram with a user specified name and type of file.


WaveFormer Pro
File
Import/Export
Edit
Bus
ParameterLibs
Project
Report
New Timing Diagram Ctrl+N
Open Timing Diagram... Ctrl+O
Merge Timing Diagram...
Compare Timing Diagram...
Save Timing Diagram Ctrl+S
Save Timing Diagram As...
Save All Files...
Print Diagram... F4
Print Parameters...
Printer Setup... Shift+F4
Exit
1 C:\SynaptiCAD\Examples\compare_test.btim
2 C:\SynaptiCAD\Examples\tbread.btim
3 C:\SynaptiCAD\Examples\text_box_images.btim
4 C:\SynaptiCAD\Examples\tutadv11.btim
5 C:\SynaptiCAD\Examples\tutadv21.btim
6 C:\SynaptiCAD\Examples\tutsim.btim
7 C:\SynaptiCAD\Examples\adv13.btim
8 C:\SynaptiCAD\Examples\analog.btim

- **Save All Files** saves all modified open files.
- **Print Diagram** either prints or creates and image file of the timing diagram window. See Section 7.5:Printing 109.
- **Print Parameters** either prints or creates and image file of the parameter table window. See Section 7.5:Printing 109.
- **Printer Setup** changes the default printer settings for this program.
- **Exit** closes the program. You will be prompted to save any unsaved files.
- The File recall list lets you quickly open timing diagram files that you have recently looked at.

Import/Export Menu

- **Import Timing Diagram From** reads waveform data from a variety of sources like VHDL/ Verilog simulators, logic analyzers, and many other EDA and test equipment sources. It then converts this data into the SynaptiCAD timing diagram format and displays the information in a timing diagram window. See Section 11.2 Import General Instructions 150.
- **Set Filter Diagram File for Import** specifies a filter file that will limit subsequent imports or timing diagrams opens so that only the signals with matching names will be loaded. A filter file can be used to control what signals get imported from a waveform file (e.g. a VCD file), the order in which they get imported, and the pod/pin mappings of the signal to a pattern generator. See Section 11.2 Import General Instructions 150.
- **Export Timing Diagram As** opens a file dialog that remembers the last type of file that you exported and allows you to save the timing diagram to different file types. See Section 11.3 Export General Instructions 154.
- **Export and Display Temporal Expression Results** creates a file callled ***diagramname* _pslresults.txt** that contains a list of the times at which temporal expressions match or fail to match. See the Transaction Tracker manual.


Import/Export
Edit
Bus
ParameterLibs
Project
R
Import Timing Diagram From...
Set Filter Diagram File for Import...
Export Timing Diagram As...
Export and Display Temporal Expression Results
Set Clocking Signal for Export...
Exclude Selected From Export
Show Non-Exported Signals...
Add Tri-State Enables for Selected Signals
Edit Pods...
Edit Object Properties ...
Add/Execute Script...

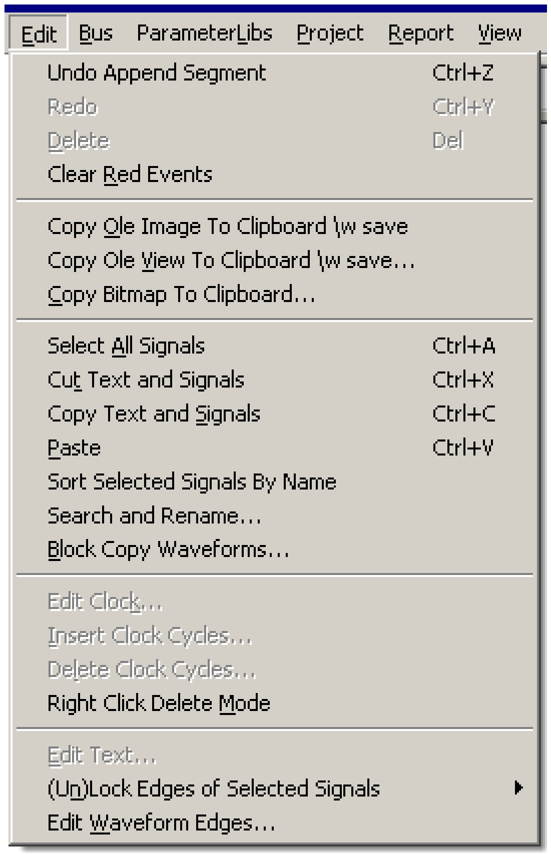
- **Set Clocking Signal for Export** opens a dialog that lets you specify the clocking signal that is used for clocked export formats like all the pattern generator formats and some third party formats. During export all of the signal values will be sampled on the positive edge of the clock and then written to the file. See Section 11.2 Import General Instructions 150.
- **Exclude Selected from Export** un-checks the **export** box in the *Signal Properties* dialog for all the signals that are currently selected. This will cause the signals to be excluded from export operations.
- **Show Non-Exported Signals** opens a dialog that 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 Tri-state Enables for Selected Signals** creates a tri-state enable signal for any signal in the timing diagram that has at least one tri-state area in it's waveform. (Used for exporting to pattern generators that do not accept tri-state values on import.) See Section 11.7 Agilent Pattern Generator -Export 168.
- **Edit Pods** opens a dialog that allows you to map signals to test equipment pods. This is used by most of the pattern generator formats, see Section 11.4 Map Signals to Test Equipment Pins 157 for general information and the special section for your particular piece of equipment.
- **Edit Object Properties** opens a dialog that lets the user attach a strings to the current timing diagram. This is an advanced feature that is used by designers who are writing their own scripts for WaveFormer Pro, DataSheet Pro or TestBencher Pro. See Section C.5 Object Properties 195.

- **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 C.6 Adding New Scripts 196).

Edit Menu

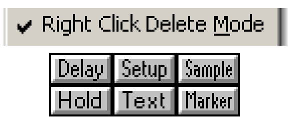
43

- **Undo *action***: Will undo last action.
- **Redo *action***: Will redo the last undo *something*
- **Delete:** Deletes a 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. See Section 3.2 Group Buses 53.


Edit
Bus
ParameterLibs
Project
Report
View
Undo Append Segment Ctrl+Z
Redo Ctrl+Y
Delete Del
Clear Red Events
Copy Ole Image To Clipboard \w save
Copy Ole View To Clipboard \w save...
Copy Bitmap To Clipboard...
Select All Signals Ctrl+A
Cut Text and Signals Ctrl+X
Copy Text and Signals Ctrl+C
Paste Ctrl+V
Sort Selected Signals By Name
Search and Rename...
Block Copy Waveforms...
Edit Clock...
Insert Clock Cycles...
Delete Clock Cycles...
Right Click Delete Mode
Edit Text...
(Un)Lock Edges of Selected Signals
Edit Waveform Edges...

- **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). See Section 7.4 OLE-Object Linking and Embedding 107.
- **Copy OLE View To Clipboard w/ save** opens a dialog that allows you to select the specific View 103 you would like to copy as an OLE object to the clipboard (Available with OLE Option for Windows). See Section 7.4 OLE- Object Linking and Embedding 107 and Section 7.3 Images and Word Processors 106.
- **Copy To Clipboard** copies the current timing diagram to the clipboard as a bitmap. See Section 7.3 Images and Word Processors 106.
- **Select All Signals** selects all signals in the active timing diagram.
- **Cut Text and Signals** removes the selected signals or text and places it on the clipboard.
- **Copy Text and Signals** copies the selected signals and any attached parameters or text to the clipboard. See Section 1.7 Copy Signals or Waveforms 31.
- **Paste Signals** pastes the signals from the clipboard back into the drawing. See Section 1.7 Copy Signals or Waveforms 31.

- **Sort Selected Signals by Name** rearranges the signals alphabetically in the diagram window.
- **Search and Rename** allows signal names to be globally searched and replaced in the diagram based on a regular expression. When a file is imported from a simulator or logic analyzer, the signal names sometimes have extra information, like hierarchical model names, that need to be striped away before exporting to the next format. See Section 11.1 Signal Names for export and import 149.
- **Block Copy Waveforms** copies sections of waveforms and paste those sections either onto (overwrite) or into (insert) any signal in the diagram. See Section 1.7: Copy Signals or Waveforms 31.
- **Edit Clock** opens the *Clock Properties* dialog for the selected clock, the same as double clicking on the waveform.
- **Insert Clock Cycles** inserts clock cycles to the right of the selected clock edge. See Section 2.2: Insert Clock Cycles 43.
- **Delete Clock Cycles** deletes clock cycles to the left of the selected clock edge. See Section 2.2: Insert Clock Cycles 43.
- **Right-click Delete Mode,** when checked, allows deleting of objects with two mouse clicks: (1) left click to select the object, and then (2) right-click to delete the object. Also notice that the other right click mode buttons will not be active. If you make a mistake, the **Edit > Undo** menu options can return the diagram to its original state.

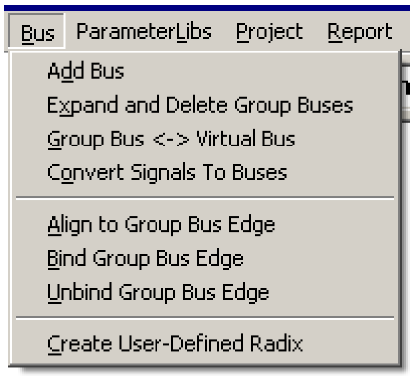


- **Edit Text** will open the *Edit Text* dialog for the selected text object (same as double clicking on the text). This dialog can change the text, font, color, size, formatting, and attachment of the text object. See Section 6.1: Adding Text 91.
- **(Un)Lock Edges of Selected Signals** either locks or unlocks all the edges of the signals which are selected.
- **Edit Waveform Edges** allows you to modify all edges on a selected set of signals according to an equation you provide. See Section 1.6 Shifting, Clearing, and Moving Edges 30.

Bus Menu

Chapter 3: Buses and Differential Signals 51 covers all of the bus functions and the differences between group, virtual, and simulated busses.

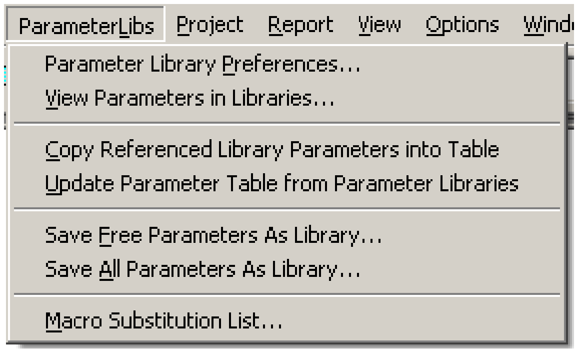
- **Add Bus** either creates a group bus from the selected signals, or if no signals are selected than it opens the *Add Bus* dialog which allows the creation of group, virtual and simulated buses.



- **Expand and Delete Group Bus** deletes a group bus and shows any member signals that may have been hidden.
- **Group Bus <-> Virtual Bus** converts the selected group bus or virtual bus to the other form.
- **Convert Signals to Buses** combines single-bit signals with matching names into buses.
- **Align to Group Bus Edge** moves all the nearby member signal transitions to the exact time of the selected bus edge.
- **Bind Group Bus Edge** binds all member signal transitions that occur at the same time as the selected bus edge to the bus edge so that if any of the transitions are moved the others will also move.
- **Unbind Group Bus Edge** removes 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 specify easily recognizable text to be displayed in place of the numerical value held by a state.

ParameterLibs Menu

The Parameter Library features are covered in Chapter 10: Parameter Libraries 138 in the Timing Diagram Editors manual.


ParameterLibs
Project
Report
View
Options
Wind
Parameter Library Preferences...
View Parameters in Libraries...
Copy Referenced Library Parameters into Table
Update Parameter Table from Parameter Libraries
Save Free Parameters As Library...
Save All Parameters As Library...
Macro Substitution List...

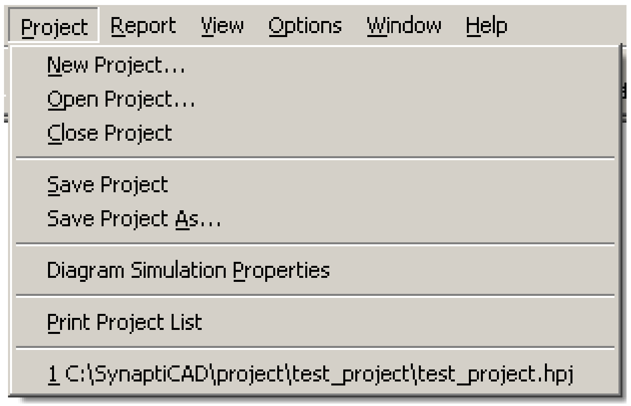
- **Parameter Library Preferences** opens the *Parameter Library Preferences* dialog in the ***Edit Parameter Libraries*** radio button so that libraries and library specifications can be added to the current project. See Section 10.1 Adding Libraries, Specifications, and Macros 138.
- **View Parameters in Libraries** opens the *View Library Parts* dialog and allows the user to view and reference parameters contained in the libraries on the library search list. See Section 10.2 Referencing and Viewing Library Parameters 140.
- **Copy Referenced Library Parameters** into Table copies the library free parameters that are currently being referenced into the timing diagram. See Section 10.5 Self-Contained Timing Diagrams 146.
- **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. See Section 10.5 Self-Contained Timing Diagrams 146.
- **Save Free Parameters as Library** saves *only the free parameters* to a file. See Section 10.3 Making Parameter Libraries 142.
- **Save All Parameters as Library** saves all the *parameters* and *free parameters* to a file (no

waveforms or signals are saved). See Section 10.3 Making Parameter Libraries[142].

- **Macro Substitution List** opens the *Edit Formula Macros* dialog and lets you define macros to replace the library specifications. See Section 10.1 Adding Libraries, Specifications, and Macros[138].

Project Menu

This describes the project window for DataSheet Pro and WaveFormer Pro with Reactive Test bench generation option. If you are using TestBencher Pro, VeriLogger, or BugHunter, then please read the project sections in manuals for those projects. This only shows the most simplest of the project features.



- **New Project** closes all open timing diagrams and creates a new project file.
- **Open Project** will open an existing HPJ file.
- **Close Project** closes the current project. You will be prompted if changes are detected.
- **Save Project** saves the current project to the same name.
- **Save Project As** saves the current project to a new name.
- **Diagram Simulation Properties** opens a dialog that controls what simulator and simulator options to use when simulating diagrams. See Section 4.5: HDL Code Generation Settings[69].
- **Print Project List** prints the names of the files that have been added to the project.

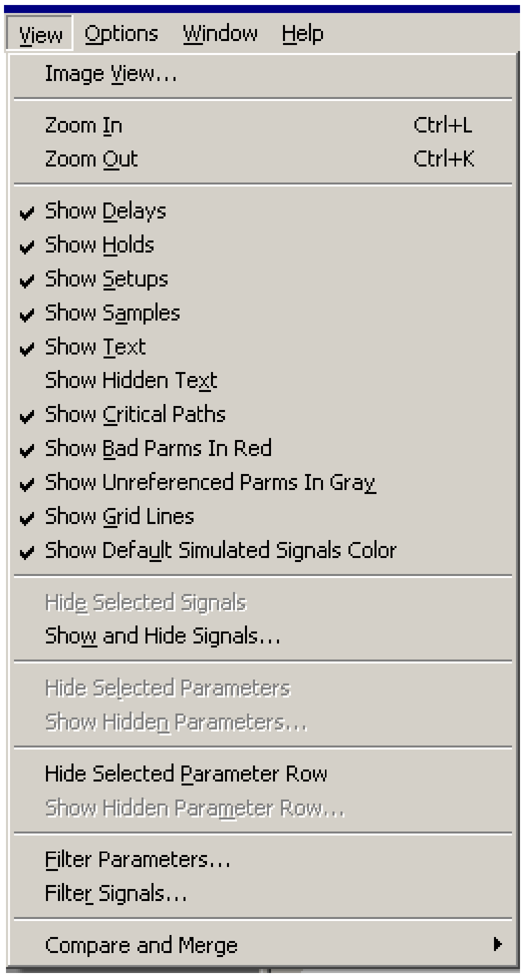
Report Menu

The Report Window editing controls are covered in Section 4.6: Report Window[72].

- **Open Report Tab** opens a new file
- **Close Report Tab** closes the displayed tab and file
- **Save Report Tab** saves the displayed file
- **Save Report Tab As** saves the displayed file under a different name.
- **Print Report Tab** prints the displayed file.

View Menu

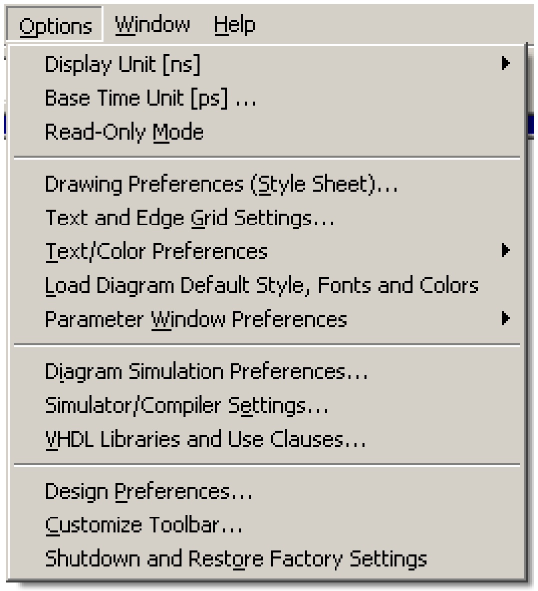
- **Image View:** Opens the *Image View Capture* dialog to create Views of the diagram for fast scrolling and printing of different sections. (Section 7.1: Views in DataSheet Pro[103]).



- **Zoom In** and **Zoom Out** menus are one way to change the level of detail in the diagram window. See Zoom Buttons 105.
- The **Show *object*** menus, if checked, show those kinds of objects.
- **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 (Chapter 1.4: Display Settings for Signals 22).
- **Show Critical Paths:** If checked, delay objects 75 are color coded to indicate which edges of the delayed transition are set by the delay.
- **Show Bad Parms in Red:** If checked, setups and holds 79 whose times have been exceeded will be drawn in red.
- **Show Unreferenced Parms in Gray**: If checked, the background of rows in the Parameter window 85 is gray for parameters which are not used in any other parameter formulas.
- **Show Default Simulated Signals Color**: If checked, simulated signals are drawn in purple.
- The **Hide Selected *object*** and **Show Hidden *object*** selectively hide and show those objects. "Parameters" are instances in the diagram window, and "Parameter rows" are the data rows in the parameter window.
- The **Filter *object*** menus allow hiding and showing to be based on a filter pattern.
- **Compare and Merge:** Opens a sub-menu that allows you to compare all signals or to search for a specific difference between the signals. See Chapter 9: Waveform Comparison 129.

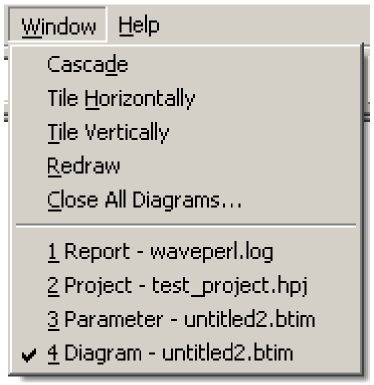
Options Menu

- **Display Unit [*ns*]** sets the unit at which time values are entered and displayed in the editor. See Section 1.10: Base and Display 36.
- **Base Time Unit [*ps*]** sets the smallest amount of time that can be displayed in the timing diagram editor. See Section 1.10: Base and Display 36.
- **Read-Only Mode**, if checked, turns off waveform editing the diagram to prevent accidental changes to the diagram. See Section 11.2 Import General Instructions 150.


Options
Window
Help
Display Unit [ns]
Base Time Unit [ps] ...
Read-Only Mode
Drawing Preferences (Style Sheet)...
Text and Edge Grid Settings...
Text/Color Preferences
Load Diagram Default Style, Fonts and Colors
Parameter Window Preferences
Diagram Simulation Preferences...
Simulator/Compiler Settings...
VHDL Libraries and Use Clauses...
Design Preferences...
Customize Toolbar...
Shutdown and Restore Factory Settings

- **Drawing Preferences** opens a dialog that controls how signals, parameters, and text are drawn: Affects how objects appear in the drawing window. See Section 1.4 Display Settings for Signals 22 and 5.3 Display Settings for Parameters 80.
- **Text and Edge Grid Settings** opens the *Edit Text and Edge Grids* dialog that controls the invisible grid that drawn edges or text snap to. See Section 6.2: Moving Text on the Alignment Grid 94 and Section 1.2: Drawing Waveforms 14.
- **Text/Color Preferences** has sub-menus to control the color or font of almost every window object in the program.
- **Load Diagram Default Style, Fonts and Colors** can return the diagram to its original style state. See Section 7.6 Miscellaneous Diagram Features 111.
- **Parameter Window Preferences** controls whether display of the min/max columns in the Parameter window show either formulas or the numerical value of the formula. See Section 5.6 Parameter Window 85.
- **Diagram Simulation Preferences** opens a dialog that controls what type of code is generated for diagram simulations (including the default settings for register and latch models). See Section 4.5: HDL Code Generation Settings 69.
- **Simulator/Compiler Settings** opens a dialog that specifies the location of external simulators or compilers to be used with the tool, and the placement of external scripts for the simulators. This is rarely used by WaveFormer customers, but is extensively used by TestBencher Pro and Bug Hunter Pro. See Section 4.5: HDL Code Generation Settings 69.
- **VHDL Libraries and Use Clauses:** opens a dialog that allows you specify library statements and uses clause that will automatically output to the VHDL testbenches. See Section 4.4 Export VHDL and Verilog test benches 65.
- **General Preferences** opens a dialog that controls how some timing analysis features work in the program. See Section 7.6 Miscellaneous Diagram Features 111.
- **Customize Toolbar** opens a dialog with check boxes to add or remove buttons from the tool bar.
- **Shutdown and restore factory settings** renames the file **syncad.ini** to **syncad.old**, then shuts down the program. When the program is restarted, all settings will be returned to their defaults.

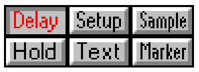
Window Menu



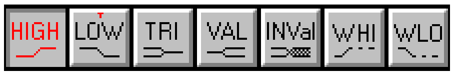
- **Cascade** layers the various windows with the active window on top.
- **Tile Horizontally** tiles the windows in "landscape" fashion (top/bottom).
- **Tile Vertically** tiles the windows in "portrait" fashion (side by side).
- **Redraw** forces the current window to completely redraw.
- **Report** brings *Report window* to the top.
- **Project** brings the *Project window* to the top.
- **Parameter** brings the *Parameter Table* to the top.
- **Diagram** brings the *timing diagram* window to the top.

Colors and their Significance

Red indicates activated buttons or error conditions.



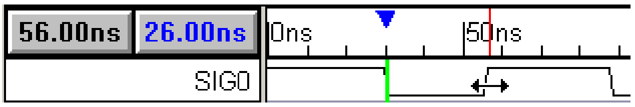
- The red **mode button** controls the type of object that the right mouse adds to the diagram. See sections 5.1 Delays 75, 5.2 Setups and Holds 79, 6.1 Text 91, 6.4 Marker 96, and 6.5 Samples 100.



- The red **state button** controls the graphical state of the next waveform segment to be drawn. See Section 1.2 Drawing Waveforms 14.
- Red **Delays** are delays that cannot draw themselves properly. Look for an error condition in the parameter window to correct the error. See Section 5.1 Delays 75
- 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. See Section 5.2 Setups and Holds 79.
- 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. See Section 2.1 Adding Clocks 39.
- Red **Waveforms** are comparison waveforms whose values differ from the companion signal. See Chapter 9 Waveform Comparisons 129.

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 Section 1.5 Measuring Time and State Values 26.



- Blue **Signal Names** with a black waveform indicates a comparison signal whose comparison did not find any differences to report. See Section 9.1 Performing a Signal Compare 130.
- Blue **Waveforms** 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 **Delays** 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 (see Section 5.1 Delays 75).

Light Gray indicates some uncertainty in the value of the object

- **Gray Waveforms** 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. These signals may also be Compare or Watch signals whose companion signal could not be found.
- **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 (see Section 5.1 Delays 75), or by double clicking on an edge and adding a minimum uncertainty to the edge (see Section 1.5 Measuring Time and State values 26).
- **Gray Delays** 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 (see Section 5.1 Delays 75).

Green indicates selection.

- **Green Delays** 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 (see Section 5.1 Delays 75).
- **Green Selection Bar** shows a selected signal transition.
- **Green dotted box** shows a selected object in the drawing window like a segment, text object, or a parameter. Select an object by Clicking on it.

Purple objects are used for simulation.

- **Purple Signal names with purple waveforms** indicate simulated signals (Chapter 4: Simulated Signals and VHDL/Verilog Export 59).

Appendix B: Writing Waveperl Scripts

Waveperl is an extended version of the Perl language that contains additional functions for manipulating data structures inside the timing diagram editor. Waveperl scripts are compiled and executed by a perl interpreter embedded into the product (technically 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 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 SynaptiCAD's 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 the product. One good example script to study is tvector.epl and its associated file **tvector.pm**. You may also find it useful to print the WaveFormer Pro 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*, 3rd Edition by Larry Wall (Perl's creator), Tom Christiansen, and Randall Schwarz.

B.1 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 and Windows environments. Perl's power and flexibility have made it a widely used language for Web-based programming. Perl has several advantages over regular programming languages:

1) Perl compilation is extremely fast for reasonable size 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. Of course, it also means you can write your own objects in Perl!

4) Perl is extensible. Many Perl modules have already been placed in the public domain that extend the capabilities of Perl, 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++.

Why a Scripting Language?

1) The built-in scripting language makes it possible for WaveFormer Pro to potentially import/export any waveform format, because the user can write a script to support it.

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.

Perl Interface

SynaptiCAD makes a large number of functions available to its embedded perl interpreter. These functions provide access to TestBencher Pro's internal data structures, and allow you to manipulate signals, delays, parameters, and the like. Many of the file export and import scripts are written entirely as perl scripts.

In the **Help** subdirectory, we include the file **twf.xs**, which is used to generate the perl functions. This file is always kept up-to-date, so you can use it as a listing of all current functions available to the perl interpreter, and their parameters. We're in the process of documenting that file, so it may be your best reference.

There are also two other files of interest: **waveperl.txt** and **twfapi.txt**. Both of these are quite out-of-date and will probably be removed or rewritten as soon as we finish documenting **twf.xs**, but until then they may provide useful information.

Finally, SynaptiCAD ships scores of perl scripts which you can examine. These scripts can be found in the **perl** subdirectory.

B.2 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 (Perl equivalent of pointers). Scalar values are automatically converted between strings and numbers depending on what the function they are passed to expects. Arrays contain an indexed list of scalar values. An associative array contains a set of key-value pairs. A value in an associative array is accessed by specifying the associated key.

Global Variables in Perl

Perl contains a number of pre-defined global variables. To distinguish these variables from user-created variables, the pre-defined 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 pre-defined variable is **$_**. $_ is the default scalar upon which many functions automatically operate if the functions are called without parameters.

For example, the statement:

```
$I = <STDIN>;
```

reads a line of characters from standard input into the variable $I. If the result of <STDIN> wasn't assigned to $I:

```
<STDIN>;
```

then **$_** would be assigned the line of characters.

The second most important pre-defined 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.

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 surround 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 '0x1111', the string literal:

```
"ADDRESS <= $state after 10 ns"
```

would be translated by Perl to:

```
'ADDRESS <= 0x1111 after 10 ns'
```

B.3 Pattern Matching (Regular Expressions)

Regular expressions are the most common way to search and change text (strings) in Perl. Regular expressions are patterns that a string can be matched against. They are commonly used in many programs.

The simplest regular expression (or regex) is just a string of alphanumeric text. This will match anything that contains that text anywhere. For example, the regex 'SIG' will match 'SIG0', 'SIGNAL', or 'ASSIGN'.

Certain characters have a special meaning in a regular expression. Here's a table explaining the most common ones:

.	This is a stand-in for any character. For example, 'SIG.' will match 'SIG0' or 'SIG1', but not something that ends in 'SIG' (since it doesn't have any characters after the letter G).
[ ]	Defines a character class, which is a group of character that a regular expression will match any one of. 'p[aeiou]t' will match 'pat', 'pet', 'pit', 'pot', or 'put', but not 'pout' since it only matches one character.
-	Specifies a range of characters within a character class. '[h-o]' is equivalent to '[hijklmno]', and '[a-zA-Z]' will match any single uppercase or lowercase letter.
^	Putting this at the beginning of a character class inverts its meaning: the regular expression will match any character not in the set. 'si[^g]' will match 'silo' but not 'signal'.
*	Allows the preceding character (or character class) to be repeated any number of times, including zero. 'gro*ve' matches 'grve', 'grove', 'groove', or 'grooooove'.
+	Allows the preceding character (or character class) to be repeated any number of times, not including zero. 'es+h' will match 'mesh' and 'governessship' but not 'beehive'.

?	Makes the preceding character (or character class) optional: it can be found either zero or one times. 'b[ou]+y' matches 'by', 'boy', and 'buy', but not 'buoy'.
^	In addition to character class inversion, this matches the very beginning of a string. '^S' will match 'SIG0', but '^I' won't.
$	This matches the very end of a string. '0+$' matches 'SIG0', 'BUS 200', and 'CLK 3000', but not 'CLK 102'.
\	Use this before a special character to remove its meaning and treat it literally. 'p\.S' will match 'top.SIG0'. You can match a literal backslash with '\\'.

Complete documentation of Perl's regular expression engine is beyond the scope of this manual; Perl's documentation contains a great deal of information 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.

Here are a few web pages with more extensive information about regular expressions:

http://search.cpan.org/dist/perl/pod/perlre.pod#Regular_Expressions

http://www.regular-expressions.info/

http://www.developer.com/lang/article.php/3330231

http://www.ternent.com/tech/regexp.html

Note: Often, a complex regular expression is far easier to code than it is to read and interpret it later, so it's a good idea to document the purpose of a regular expression.

B.4 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 **Import/Export > Export Timing Diagram 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 gets printed to standard output (the export file specified by the user).

Tips on Debugging Waveperl Scripts

Error messages during compilation or execution of a Waveperl script are redirected to a file called **waveperl.log**. The contents of this file is automatically displayed in a tab in the Report Window. When debugging a new script, it is a good idea to constantly watch this file!

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 **Import/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 does significantly slow down initial script execution speeds (they get recompiled each time they are run).

Perl and WaveFormer Pro API Documentation

Perl: Before you write your first script, we recommend you visit http://www.perl.com if you haven't programmed in Perl before. There are also many books on Perl available in most bookstores. Next, you will want to read the WaveFormer Pro API documentation. Finally, 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.

WaveFormer Pro API Introduction: The file **twfapi.txt** contains a description of common functions in WaveFormer Pro that can be called from Waveperl scripts.

Sharing your scripts with others: If you write a script and you think it may be useful to other users, we encourage you to email it to us at **sales@syncad.com** so that it can be shared with other users. 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.

Perl API Commands

The most up-to-date docs are in the two "xs" files located in **C:\SynaptiCAD** directory. These list all Perl API calls. The **btimtwf.xs** file lists the core routines for handling timing diagram document information. The **twf.xs** file has all the rest of the commands (e.g. commands for displaying message boxes, prompting for data, managing windows, detecting currently selected objects, etc.).

B.5 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 testbench).

To add a property to an object in the timing diagram:

- Select the **Import/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 by choosing the the **Import/Export > Edit Object Properties** menu which opens a dialog with the property (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 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**).

B.6 Adding New Scripts

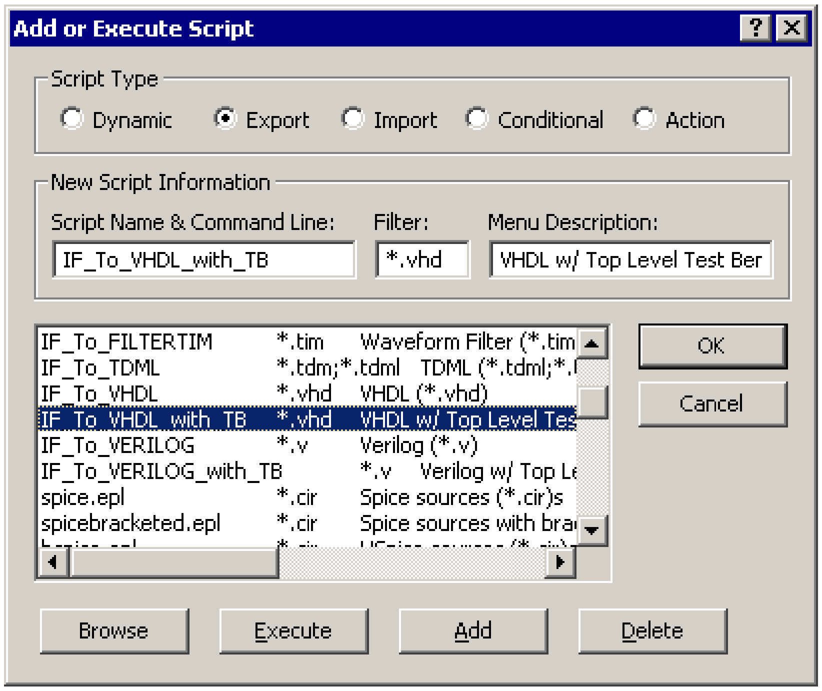
There are several types of scripts that perform different types of actions. By convention the name of the script indicates the type of script.

- **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.
- **Import Script** names end with ".ipl". They read an external file and convert that information into objects in the current timing diagram.
- **Dynamic Script** names end with the extension ".pl". They perform functions on the current timing diagram.
- **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 types scripts. These scripts describe different conditions and actions that a Sample parameter can trigger on or act on inside the timing diagram editor.

Adding an new Script

When you design your own script, you will have to tell WaveFormer Pro that it exists. This will ensure that it appears in either the **Save File as Type** list box of the *Export* dialog (export script), or the **List Files of Type** list box of the *Open* dialog (import script), or the **Export** menu (dynamic script). To add a script to WaveFormer Pro:

- Select the **Import/Export > Add or Execute Script** menu option. This will open a dialog of the same name.



Script Type

( ) Dynamic [✓ Export] ( ) Import ( ) Conditional ( ) Action

New Script Information

Script Name & Command Line: IF_To_VHDL_with_TB

Filter: *.vhd

Menu Description: VHDL w/ Top Level Test Ber

IF_To_FILTERTIM	*.tim	Waveform Filter (*.tim
IF_To_TDML	*.tdm;*.tdml	TDML (*.tdml;*.t
IF_To_VHDL	*.vhd	VHDL (*.vhd)
IF_To_VHDL_with_TB	*.vhd	VHDL w/ Top Level Tes
IF_To_VERILOG	*.v	Verilog (*.v)
IF_To_VERILOG_with_TB	*.v	Verilog w/ Top Le
spice.epl	*.cir	Spice sources (*.cir)s
spicebracketed.epl	*.cir	Spice sources with bra

- Choose the type of script to be added: Dynamic, Export, or Import.
- Type the script name in the **Script Name & Command Line** edit box. By convention, dynamic script names end with the extension ".pl", export script names end with ".epl", and import script names end with ".ipl".
- 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.
- Click the **ADD** button to add the new script to the list.
- Click the **OK** button to close the dialog.

Appendix C: WaveFormer Lite Help

WaveFormer Lite is a special version of WaveFormer Pro that can generate VHDL and Verilog stimulus-based test benches for the Actel Libero design software and other FPGA/ASIC vendor flows. WaveFormer Lite fits seamlessly into the Actel's design environment, automatically extracting signal information from your HDL design files, and producing HDL test bench code that can be used with any standard VHDL or Verilog simulator. WaveFormer Lite is no longer shipped by Actel as part of the Libero package, but it can be purchased from SynaptiCAD and will still work with Libero.

The latest version of WaveFormer includes a built-in project window, allowing you to create testbenches for any VHDL/Verilog design regardless of what design flow tools you use.

- Section C1: WaveFormer Lite Design Flow 198 has instructions for generating VHDL and Verilog with the Actel design environment.
- Section C2: WaveFormer Lite Upgrade Options 199 describes how to upgrade to the professional version of WaveFormer Pro. The professional version has features that help you create timing diagrams faster, analyze circuit timing, produce Data Book quality images, and translate waveform information from over 35 formats including most popular logic analyzers, pattern generators, and simulators.
- Section C3: Specifying Signal Types for Actel Fusion Analog Signals 201 describes how to set the Analog types for Fusion Analog Signals

C.1 WaveFormer Lite Design Flow

WaveFormer Lite generates VHDL and Verilog test benches from drawn waveforms. There are three basic steps for creating test benches using WaveFormer Lite and the Actel design software:

1) Import and setup the signal information:

- If you are not using Libero then select **Project > New Project** menu and add your source code files to the list as shown in Section 4.4 Export VHDL and Verilog test benches 65.
- If you cannot see the WaveFormer Icon inside Libero, then you must set the Profiles to point to WaveFormer Lite. First right click and choose **stimulus** from context menu and select **Profiles**. The add a new **Stimulus Profile** and point it to the new **Syncad.exe** file.
- Follow the Actel instructions for launching WaveFormer Lite. The first time, the Actel software will create a project and hand it to WaveFormer Lite which will extract all of the signal information from the top-level ports. WaveFormer Lite will then create a timing diagram containing all of the signals (including the type, direction and size).
- If you later change the ports of the Model Under Test, you can force a signal extraction by clicking the **Extract MUT Ports into Diagram** button located on the program level button bar
- **(Fusion Only)** The export type for each analog signal needs to set according to the instructions in Section C3: Specifying Signal Types for Actel Fusing Analog Signals 201.

2) Draw the Test Bench:

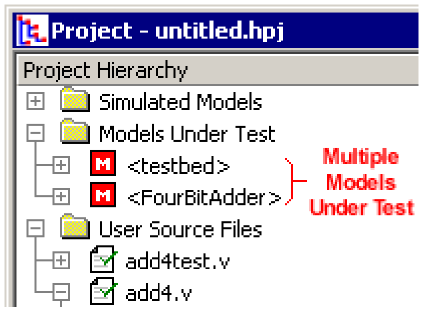
- Draw the waveforms on the signals to describe the testbench. Section 1.2 Drawing Waveforms 14 describes how to use the mouse and the state buttons to draw waveforms.
- For clock signals, right click on the clock name and choose **Signal(s) <=> Clock(s)** menu to

convert the signal to a SynaptiCAD clock which will draw its own waveform based on the period and frequency. Double click on the waveform to edit the clock properties. Section 2.1 Adding Clocks 39 describes all of the clock features.

- Add **Analog Waveforms** using Python equations as shown in Section 8.2 Waveform Equation Blocks for editable Analog waveforms 116.
- (optional) **Add Reactive Test Bench objects:** Libero Platinum users used to receive Reactive Test Bench features with WaveFormer Lite that enabled them to generate test benches that can verify and react to output from the model under test. WaveFormer Lite with Reactive test bench generation can be purchased from SynaptiCAD directly. For more information on these capabilities, see Reactive Test Bench Manual on the Help menu. These users will be adding samples to the expected inputs to the testbench to indicate the times at which to test the values output by the model under test.

Changing the Model Under Test:

- The **Extract MUT Ports into Diagram** function makes a guess as to which model is the model under test and displays that model with single brackets, <>, underneath in the **Models Under Test** folder.



- To pick a different model under test, first right click on the MUT and choose **Unset Current Model Under Test**, and then right click on a different model under the *User Source Files* list and pick **Set as Model Under Test**. Multiple models under test can also be specified.
- Then press **Extract MUT Ports into Diagram** button to re-populate the *Stimulus and Results* diagram.

3) Export the Test Bench:

- Select the **Export > Export Timing Diagram** menu to open the file dialog, and in the **save as type** box choose either the **VHDL w/ top level test bench** or **Verilog w/ top level test bench**. The **Top Level Test Bench** types instantiates the model in the *Project* window and the stimulus model within a top-level model. This top-level module can then be simulated in Actel Libero without any additional setup.
- Closing this dialog generates the test bench and also displays it in the *Report* window so that you view the generated code.
- Section 4.4 Exporting VHDL and Verilog test benches 65 has more information about exporting and also about adding VHDL libraries and use clauses.

C.2 WaveFormer Purchasing Options

WaveFormer Lite is a reduced version of WaveFormer Pro that is specifically made to generate test benches for Actel's Libero software. Libero can call WaveFormer Lite and had it a list of signals of the ports of the model under test. Inside WaveFormer Lite, users draw the testbench waveforms and then WaveFormer Lite generates the test bench. WaveFormer Lite does not include the timing analysis features of the full Pro version. You can also add the Reactive Test Bench Option that will create test benches that respond to the model-under-test during the simulation.

WaveFormer Pro is a professional version of WaveFormer Lite that you can purchase separately from SynaptiCAD to use with your Actel design software. WaveFormer Pro has many features that help you create timing diagrams faster, analyze circuit timing, produce high quality images, and translate waveform information from over 35 formats including most popular logic analyzers, pattern generators, and simulators. Here are just a few of the major features included in WaveFormer Pro:

- An Interactive Simulator 59 lets you describe waveforms using Boolean and registered logic equations instead of having to draw them manually.
- Delay, Setup, and Hold parameters 75 allow you to document the causal relationships between the waveform edges making the timing diagrams more readable.
- Logic Analyzer and Pattern Generator support 148 lets you move test data back and forth between the simulation and hardware prototyping environment. You can capture an existing hardware interface using a logic analyzer and use WaveFormer Pro to translate the data into a VHDL or Verilog testbench. And WaveFormer Pro can take your simulation output waveforms and convert them into stimulus files for the pattern generator.
- Professional Documentation features 103 enable you to create timing diagrams that are "ready for publication" in design documentation, data books, or on web sites.
- Analog Signal Display feature 115 shows analog signals as a magnitude plot rather than the digital bus display, allowing you to view the signal as it would display on a digitizing oscilloscope. Every signal has an *Analog Display* check box in the Signal Properties dialog and the height of each signal is individually controllable. Analog signals can be displayed using either point-to-point or step-function plots.

Simulation and Test Bench Upgrades:

- **VeriLogger Pro** is a new type of Verilog simulation environment that combines all the features of a traditional Verilog simulator with the most powerful graphical test vector generator on the planet. Model testing is so fast in VeriLogger Pro that you can perform true bottom-up testing of every model in your design, a critical step often skipped in the race to market.
- **TestBencher Pro** generates VHDL and Verilog bus-functional model test benches that can be used to test large systems. In comparison, WaveFormer Lite generates simple stimulus-based test benches. TestBencher Pro automates the most tedious aspects of test bench development, allowing you to focus on the design and operation of the test bench. This is accomplished by representing each bus transaction graphically and then automatically generating the code for each transaction. TestBencher makes use of the powerful features of the language that is being generated and the engineer does not have to hand-code each transaction. When hand coding, the designer would have to take the time to deal with the specifics of the design (port information, monitoring system response, etc.) as well as common programming errors (race conditions, minor logic errors, and code design problems). This removes a considerable amount of time from the test bench design process because TestBencher manages the low-level details and automatically generates a valid test bench.

Upgrading WaveFormer Lite

After purchasing a license for another SynaptiCAD product, you will need to set up a General Preferences option so that Libero will automatically run that product instead of WaveFormer Lite:

- Install and run your new SynaptiCAD product.
- Select the **Options > General Preferences** menu option to open the *General Preferences* dialog.
- In the **Substitute for WaveFormer Lite** drop-down menu, select the product you purchased.

- Click **OK** to close the dialog. This will save a special option in the .ini file that will automatically launch your other program instead of WaveFormer Lite when you use Libero.
- Run Libero and double-click on the WaveFormer Lite icon in the project window. This will launch your new SynaptiCAD product.

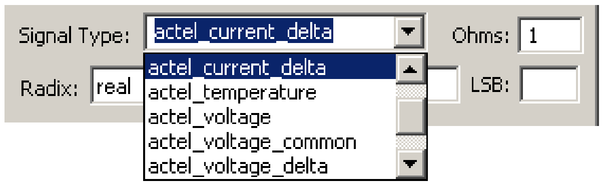
To purchase an upgrade contact SynaptiCAD at:

- Phone sales: 1-800-804-7073
- Phone tech support: 1-(540)-953-3390
- Fax: 1-(540)-953-3078
- Email: sales@syncad.com
- Web Site: www.syncad.com (the web site will also have the latest WaveFormer Lite software which is usually a few months newer than the product that Libero ships).

C.3 Specifying Signal Types for Actel Fusion Analog Signals

Analog Fusion Signals use digital HDL types of std_logic (VHDL) or wire (Verilog) in the declaration of the model under test. This means that WaveFormer cannot automatically tell whether a signal is analog or digital or part of a differential analog pair. Therefore, when working with Fusion analog signals, you must manually set the types of the signals and define the signal pairings for the differential signals.

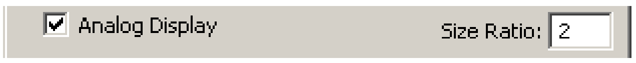
- Double click on a signal's name to open the *Signal Properties* dialog, and then select an Actel type from the **Signal Type** drop down.


Signal Type: actel_current_delta   Ohms: 1
Radix: real   LSB:
actel_current_delta
actel_temperature
actel_voltage
actel_voltage_common
actel_voltage_delta

- **actel_voltage** holds analog voltage values
- **actel_temperature** holds analog temperature values in Celsius
- To create a ***differential signal pair***, create two signals, select them, and press the **Add Bus** button to pair them. Note that one signal of the pair must have its type set to actel_voltage_common, and the other signal must have its type set to either actel_voltage_delta or actel_current_delta. The differential signal pair types are:
    - **actel_voltage_common**: Common mode voltage (any pair requires one of these)
    - **actel_voltage_delta**: Delta voltage signal
    - **actel_current_delta**: Delta current signal. When this type is chosen, an **Ohms** field will appear to the right of the type box. Use this field to specify the resistance value.

Display Waveform as an Analog Signal:

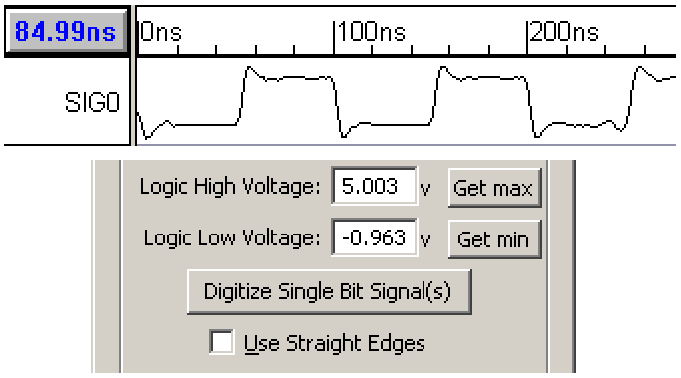
- Double-click on the signal name to open the *Signal Properties* dialog, and check the **Analog Display** box.


☑ Analog Display   Size Ratio: 2

- To increase the vertical height of the signal, set the **Size Ratio** box to be greater than 1.

Copyright © 2011, SynaptiCAD Sales, Inc.

- Section 8.1: Analog Waveform Display 115 explains how to control the voltage range and also how the radix or the MSB/LSB affect how the waveform is displayed.



Logic High Voltage: 5.003 v
Logic Low Voltage: -0.963 v
Use Straight Edges

Display Differential signals as one signal pair:

- Section 3.4 Differential Signals 56 has the instructions for creating a differential signal that shows both of the waveforms of the pair on top of each other. This differential signal is a 2-bit group bus and is for display purposes only. The member signals will be exported to the testbenches (not the group signal).

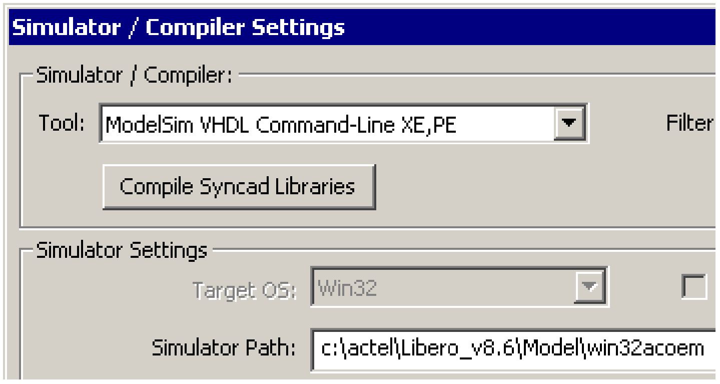
C.4 Compiling Latest ModelSim Libraries

The testbenches created by WaveFormer Lite use pre-written packages that must be compiled by ModelSim into a VHDL library called syncad_vhdl_lib. Below are the steps for creating this library and mapping the path to this library for ModelSim.

Compiling the syncad library with ModelSim (once per update of ModelSim):

Whenever you update your version of ModelSim, you will need to recompile the SynaptiCAD testbench library with the new ModelSim compiler by following the steps below:

- Start WaveFormer and select the **Options > Simulator/Compiler Settings** menu option to open the *Simulator and Compiler Settings* dialog.



Tool: ModelSim VHDL Command-Line XE,PE
Target OS: Win32
Simulator Path: c:\actel\Libero_v8.6\Model\win32acoem

- From the **Tool** drop-down box, select the your particular ModelSim VHDL compiler. If you are not sure which version of ModelSim that you have, select **ModelSim VHDL Command-Line XE/PE**.
- Set the path where your ModelSim is located. For example, if you are using a version of ModelSim from Actel Libero on Windows, the Simulator Path will look something like:

```
C:\Actel\Libero_8.6\Model\win32oem
```

- Press the **Compile Syncad Libraries** button to compile the SynaptiCAD Libraries. The first time that you set up a particular simulator or compiler you should press this button.

- You can view the results of this simulation by looking in the **simulation.log** tab of the *Report* window. If you cannot see the *Report* window, then choose the **Window > Report** menu option to bring the window to the front.

Map SynaptiCAD library to ModelSim:

Whenever you create a new project, you will need to inform ModelSim of the location of the syncad_vhdl_lib by running ModelSim's **vmap** command:

- Launch a DOS command prompt and change directory to your project's current working directory using the DOS **cd** command. Then run vmap using the options given below:

```
cd \synapticad\project\add4
vmap syncad_vhdl_lib "%APPDATA%\Synapticad\lib\vhdl\modelsim_vhdl_lib"
```

Index