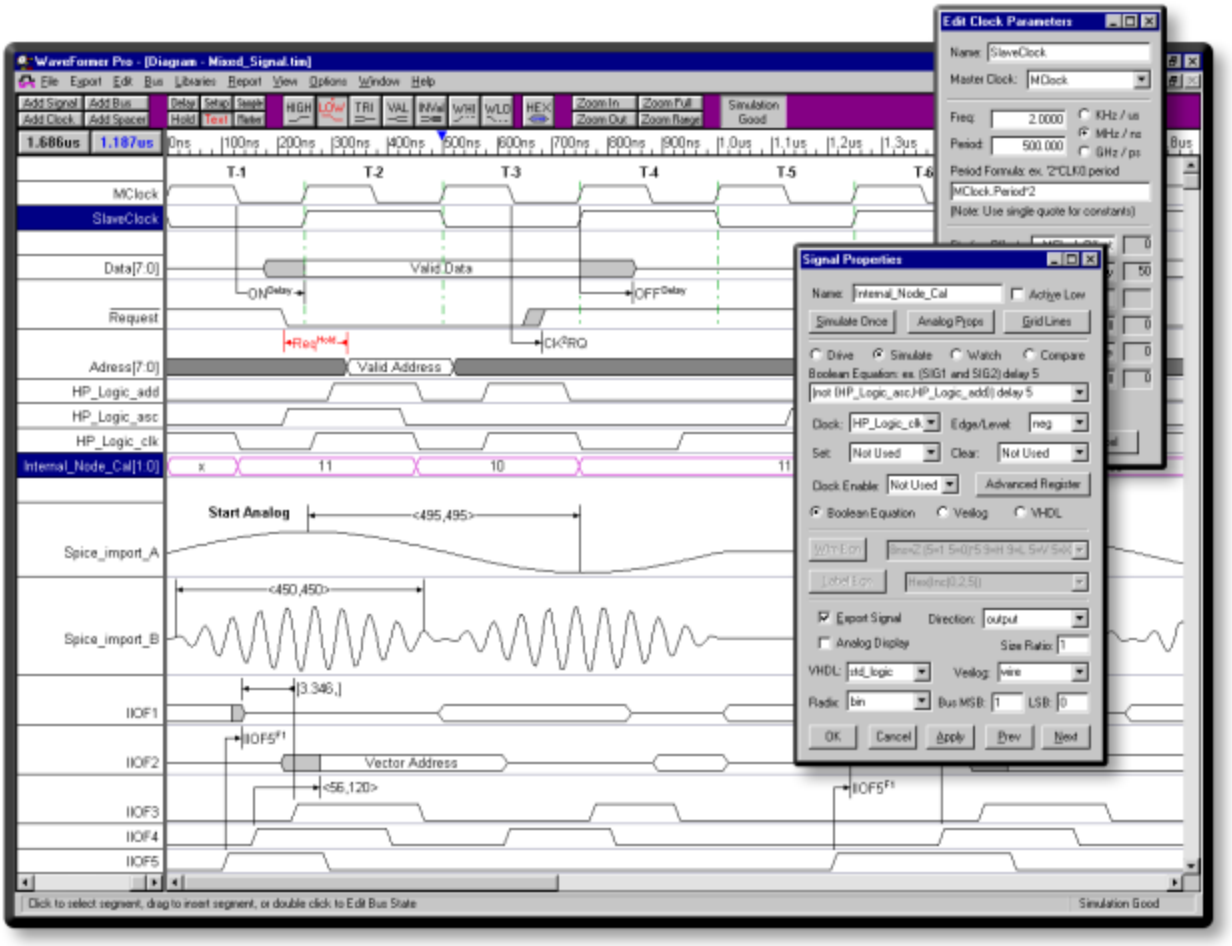
SynaptiCAD Tutorials

Copyright © 2014, SynaptiCAD Sales, Inc.



SynaptiCAD Tutorials

Copyright Copyright © 2014, 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: September 2014 in (whereever you are located)

SynaptiCAD Product Tutorials

DataSheet Pro, WaveFormer Pro, Timing Diagrammer Pro

BugHunter Pro, VeriLogger Extreme, VeriLogger Pro

TestBencher Pro, GigaWave Viewer, Transaction Tracker

SynaptiCAD's tutorials demonstrate everything from how to draw basic timing diagrams to advanced VHDL and Verilog simulation techniques. The tutorials are grouped together by product category, so that it is easy to pick the tutorials that cover the features that you need for your design. If you are new to our product line, the best tutorial to start with is the Basic Drawing and Timing Analysis tutorial, because this demonstrates how to draw waveforms and the general design strategy for our timing diagram editors. Some of the features demonstrated require additional licenses for the particular software option. Please see our web site or contact our sales department for specific information on those features.

Table of Contents

Foreword 0

Timing Diagram Editor 1: Basic Drawing and Timing Analysis 10

(TD) 1.1 Timing Diagram Editor Choices ... 10
(TD) 1.2 Set the Base and Dispaly Time Unit ... 12
(TD) 1.3. Add the Clock ... 13
(TD) 1.4 Add the Signals ... 15
(TD) 1.5 Drawing Signal Waveforms ... 15
(TD) 1.6 Editing Signal Waveforms ... 17
(TD) 1.7 Adjust Diagram to Match Figure ... 19
(TD) 1.8 Add the D Flip-Flop Propagation Delay ... 20
(TD) 1.9 Add the Inverter Propagation Delay ... 22
(TD) 1.10 Add the Setup for the Dinput to Clock ... 24
(TD) 1.11 Add a Free Parameter ... 25
(TD) 1.12 Drawing with Equations ... 27
(TD) 1.13 Drawing Virtual Busses ... 29
(TD) 1.14 Drawing Group Buses and Differential Signals ... 30
(TD) 1.15 Working with Drawing Environnment ... 33
(TD) 1.16 Summary ... 35

Timing Diagram Editor 2: Simulated Signals 36

(TD) 2.1 Setup for Simulation ... 36
(TD) 2.2 Simulate a Boolean Equation ... 38
(TD) 2.3 Boolean Equations with Delays ... 39
(TD) 2.4 Register and Latch Signals ... 40
(TD) 2.5 Set and Clear Lines ... 42
(TD) 2.6 Multi-bit Equations ... 44
(TD) 2.7 Design a Multi-Bit Counter ... 46
(TD) 2.8 End Diagram Marker Stops Simulation ... 47
(TD) 2.9 Behavioral HDL Code ... 48
(TD) 2.10 Simulated Bus Signals ... 50
(TD) 2.11 Summary of Simulated Signals Tutorial ... 52

Timing Diagram Editor 3: Display and Documentation 53

(TD) 3.1 Setup for the Tutorial ... 53
(TD) 3.2 Parameter Display Strings ... 54
(TD) 3.3 Repeat Parameters Across the Diagram ... 56

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 3.4 Move Parameters to Different Signals ... 57
(TD) 3.5 Adjust Parameter Vertical Placement ... 57
(TD) 3.6 Curved Parameters ... 58
(TD) 3.7 Clock Jitter and Display ... 59
(TD) 3.8 Marker Time Compression ... 61
(TD) 3.9 Marker Snap to Edge ... 63
(TD) 3.10 Marker Loops and Pipelines ... 64
(TD) 3.11 Spacers and Text Font Controls ... 65
(TD) 3.12 Highlight Regions with Text Objects ... 67
(TD) 3.13 Text and Hidden Signals ... 67
(TD) 3.14 Summary of Display and Documentation Tutorial ... 69

Timing Diagram Editor 4: Analog Signals 70

(TD) 4.1 Viewing Analog Waveforms ... 70
(TD) 4.2 Faster Drawing with Waveform Equation Blocks ... 73
(TD) 4.3 Writing Python Waveform Equation Blocks ... 74
(TD) 4.4 State Label Equation Alternative ... 76
(TD) 4.5 Drawing a Step Signal ... 77
(TD) 4.6 Generating Sine Waves ... 78
(TD) 4.7 Generating Capacitor Charge and Discharge ... 81
(TD) 4.8 Generating Ramp Waveforms ... 83
(TD) 4.9 Random Analog Equations ... 83
(TD) 4.10 Exporting to SPICE, VHDL, and Verilog ... 85
(TD) 4.11 ADC and DAC Conversion ... 86
(TD) 4.12 Summary of Analog Signals Tutorial ... 88

Timing Diagram Editor 5: Parameter Libraries 90

(TD) 5.1 Setup for Library Tutorial ... 90
(TD) 5.2 Add Libraries to the "Library Search List" ... 91
(TD) 5.3 Setup the Library Specifications ... 92
(TD) 5.4 Investigate Preferences Dialog ... 92
(TD) 5.5 Referencing Parameters in Libraries ... 92
(TD) 5.6 Using Macros to Examine Tradeoffs Between Different Libraries ... 95
(TD) 5.7 Parameter Libraries Summary ... 97

Timing Diagram Editor 6: Advanced Modeling and Simulation 98

(TD) 6.1 Set up a New Timing Diagram ... 99
(TD) 6.2 Generate the Clock, Draw Waveforms, & Use Waveform Equations ... 100
(TD) 6.3 Modeling State Machines ... 101

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 6.4 Checking for Simulation Errors ... 103
(TD) 6.5 Incremental Simulation ... 104
(TD) 6.6 Modeling Combinational Logic ... 105
(TD) 6.7 Entering Direct HDL Code for Simulated Signals ... 105
(TD) 6.8 Modeling n-bit Gates ... 106
(TD) 6.9 Incorporating Pre-written HDL Models into Waveformer Simuations ... 106
(TD) 6.10 Modeling the Incrementor and Latch Circuit ... 107
(TD) 6.11 Modeling Tri-State Gates ... 108
(TD) 6.12 Debugging External Verilog Models ... 108
(TD) 6.13 Verify the Histogram Circuit ... 108
(TD) 6.14 Controlling the Length of the Simulation ... 109
(TD) 6.15 Editing Verilog Source Files ... 109
(TD) 6.16 Simulating Your Model with Traditional Verilog Simulators ... 110
(TD) 6.17 Summary ... 110

Test Bench Generation 1: VHDL-Verilog Stimulus 111

(TBench) 1.1 Load the Tutorial Timing Diagram ... 111
(TBench) 1.2 Hex and Binary State Values ... 112
(TBench) 1.3 Export a Verilog Test Bench ... 113
(TBench) 1.4 Signal Data Types and VHDL user defined types ... 115
(TBench) 1.5. Export a VHDL Test Bench ... 118
(TBench) 1.6 Summary of VHDL-Verilog Stimulus Tutorial ... 121

Test Bench Generation 2: Reactive Test Bench Option 122

(TBench) 2.1 Run Program with Reactive Test Bench Option ... 122
(TBench) 2.2 Create a Project to hold the MUT ... 122
(TBench) 2.3 Extract Signal Names and setup the Clock ... 124
(TBench) 2.4 Draw Stimulus Waveforms and Export Test Bench ... 127
(TBench) 2.5 Draw Expected Waveform and Wait for the Assertion ... 129
(TBench) 2.6 Draw a Read Cycle and Verify the read ... 132
(TBench) 2.7 Add a Sample to Verify Data Read from MUT ... 133
(TBench) 2.8 Drive Waveform Values using a File ... 135
(TBench) 2.9 Create For-Loop to Perform Multiple Writes and Reads ... 138
(TBench) 2.10 Alternate Test Bench Designs ... 140
(TBench) 2.11 Summary of Reactive Test Bench Tutorial ... 141

Test Bench Generation 3: TestBencher Pro Basic Tutorial 142

(TBench) 3.1 Run TestBencher Pro ... 142

Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 3.2 Create a Project ... 143
(TBench) 3.3 Add the SRAM model to the Project ... 145
(TBench) 3.4 Setup the Template Diagram ... 146
(TBench) 3.5 Create the Write Cycle Transaction Diagram ... 148
(TBench) 3.6 Create the Read Cycle Transaction Diagram ... 150
(TBench) 3.7 Add a Sample to Verify Data ... 152
(TBench) 3.8 Create the Initialize Transaction Diagram ... 154
(TBench) 3.9 Add Transaction Calls to the Sequencer Process ... 156
(TBench) 3.10 Setup the Simulator ... 159
(TBench) 3.11 Generate the Test Bench and Simulate ... 160
(TBench) 3.12 Examine Report Window Results ... 160
(TBench) 3.13 Examine the Stimulus and Results Diagram ... 162
(TBench) 3.14 TestBencher Pro Basic Tutorial Summary ... 162

Test Bench Generation 4: TestBencher Pro with Random Transactions 163

(TBench) 4.1 Run TestBencher Pro ... 163
(TBench) 4.2 Setup the VHDL Simulator ... 164
(TBench) 4.3 Load the RandomizedSweepTest Project ... 165
(TBench) 4.4 Weight the Transaction Types ... 166
(TBench) 4.5 Post Random Transaction Types ... 167
(TBench) 4.6 Constrain the Random Data ... 168
(TBench) 4.7 Simulate and View the Results ... 170
(TBench) 4.8 Set the Random Seed ... 172
(TBench) 4.9 Randomize Transactions Summary ... 173

Simulation 1: VeriLogger Basic Verilog Simulation 174

(Sim) 1.1 Simulator Choices ... 174
(Sim) 1.2 Add Files to the Project ... 175
(Sim) 1.3 Build the Tree and Investigate the Project ... 177
(Sim) 1.4 Simulate the Project ... 180
(Sim) 1.5 Prepare for Graphical Test Bench Generation ... 181
(Sim) 1.6 Draw Test Bench in Debug Run Mode ... 183
(Sim) 1.7 Simulate in Auto Run Mode ... 185
(Sim) 1.8 Breakpoints, Stepping and Inspecting ... 187
(Sim) 1.9 Archiving Stimulus and Results ... 189
(Sim) 1.10 Saving the Project files ... 190
(Sim) 1.11 Summary of VeriLogger Basic Verilog Simulation ... 191

Simulation 2: Using WaveFormer with ModelSim VHDL 192

Copyright © 2014, SynaptiCAD Sales, Inc.

(Sim) 2.1 Compile SynaptiCAD Library Models ... 192
(Sim) 2.2 Create a project and extract the ports ... 194
(Sim) 2.3 Draw the test bench waveforms ... 196
(Sim) 2.4 Export Waveforms to VHDL ... 198
(Sim) 2.5 Simulate VHDL test bench using ModelSim ... 199
(Sim) 2.6 Compare simulation results against expected results ... 200
(Sim) 2.7 Summary of Using WaveFormer with ModelSim VHDL ... 204

Waveform Comparison Tutorial 205

(Compare) 1: Setup for using Compare ... 205
(Compare) 2: Individual Compare Signals ... 206
(Compare) 3: Experiment with Tolerance ... 208
(Compare) 4: Compare Timing Diagrams ... 209
(Compare) 5: Set All Compare Signal Properties ... 211
(Compare) 6: Find the Differences ... 213
(Compare) 7: Perform a Clocked Comparison ... 215
(Compare) 8: Compare During Clock Cycle Windows ... 217
(Compare) 9: Mask Sections to Exclude Comparison ... 219
(Compare) 10: Don't Care Regions ... 221
(Compare) 11: Adjust the Time Difference Between Two Diagrams ... 222
(Compare) 12: Summary of the Comparison Tutorial ... 223

Gigawave and WaveViewer Viewer Tutorial 225

(Viewer) 1: Converting a vcd file into a btim file ... 225
(Viewer) 2: Importing a subset of the Waveforms ... 226
(Viewer) 3: Creating a Filter File to selectively load signals ... 228
(Viewer) 4: Show and Hide Signals in the display ... 230
(Viewer) 5: Searching for Signal Names or States ... 230
(Viewer) 6: Options: Gigawave, Waveform Comparison,Transaction Tracking ... 232
(Viewer) 7: Waveviewer/GigaWave Viewer Tutorial Summary ... 234

SDC Timing Generation Tutorial 235

(SDC) 1: Setup Project and Create Timing Diagram ... 235
(SDC) 2: Enter Master Clocks ... 236
(SDC) 3: Enter Derived Clocks ... 238
(SDC) 4: Sketching the Waveforms ... 239
(SDC) 5: Signal Direction ... 240
(SDC) 6: Add Delays to Inputs ... 241
(SDC) 7: Add Setups and Holds to Outputs ... 243

Copyright © 2014, SynaptiCAD Sales, Inc.

(SDC) 8: VCD Simulation Files Generate SDC 246
(SDC) 9: Summary 248

Transaction Tracker Tutorial 249

(TT) 1: Open the Example File 250
(TT) 2: Match all occurrences of a simple pattern 250
(TT) 3: Match Consecutive Occurrences with Concatenation Operator 251
(TT) 4: Match with consecutive repetition Operator 251
(TT) 5: Match with non-consecutive Repetition Operator 252
(TT) 6: Bit-slices and the Boolean operators 252
(TT) 7: Implication operator 253
(TT) 8: Implication Next-Cycle operator 253
(TT) 9: PSL Property 253
(TT) 10: Summary of Transaction Tracker Tutorial 254

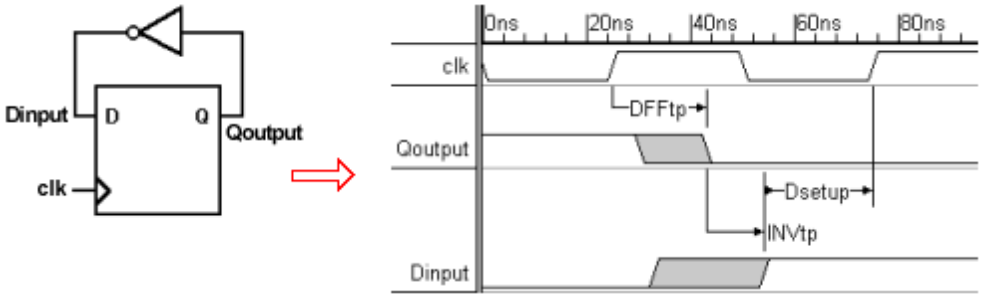
Index 255

Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 1: Basic Drawing and Timing Analysis

This tutorial demonstrates the basic timing diagram editor features. It teaches you how to draw timing diagrams using delays, setups, clocks and part libraries and how to use timing diagrams to help detect timing errors in digital designs. It also covers the waveform editing features, measurement and quick access buttons.

You will draw the timing diagram for a circuit that divides the clock frequency in half. Both the flip-flop and the inverter have propagation times that delay the arrival of the Dinput signal. If Dinput is delayed too long it will violate the data-to-clock setup time (Dsetup). This increases the risk of the flip-flop failing to clock in the data and may lead to the flip-flop entering a metastable state.



clk	20MHz	(50ns period)
DFFtp	5-18ns	D flip-flop (74ALS74): Clock to Q propagation time
Dsetup	15ns minimum	D flip-flop (74ALS74): D to rising edge Clock setup time
INVtp	3-11ns	Inverter (74ALS04): propagation time

Above is an image of the timing diagram and parameter table that you will enter during the tutorial. The first thing you may notice is the gray signal transitions caused by the min/max values of the component delays. The gray areas of the signal transitions are uncertainty regions, which indicate that the signal may transition any time during that period. This is a little disconcerting especially if you have been using a low-end simulator that cannot compute both min and max at the same time. This representation shows the entire range of possible circuit performance, so that there won't be any surprises during production when you get components at extreme ends of their tolerance range.

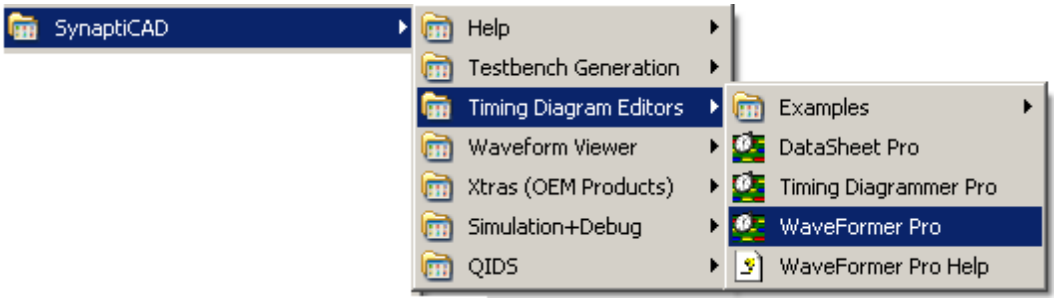
(TD) 1.1 Timing Diagram Editor Choices

SynaptiCAD has three levels of timing diagram editors. The most basic is **Timing Diagrammer Pro**, which allows drawing and basic timing analysis using delays, setups and holds. The middle level is **WaveFormer Pro**, which has a built in simulation engine that allows signals to be described using Boolean and registered logic equation. WaveFormer is also a universal waveform translator and can take waveforms from one format and convert it to a different format. And the highest level is **DataSheet Pro** which supports multiple timing diagram display, object linking and embedding, and more image file formats. Each editor has all of the features of the products that are below it. This tutorial covers basic features which are supported in all three editors.

Run a Timing Diagram Editor:

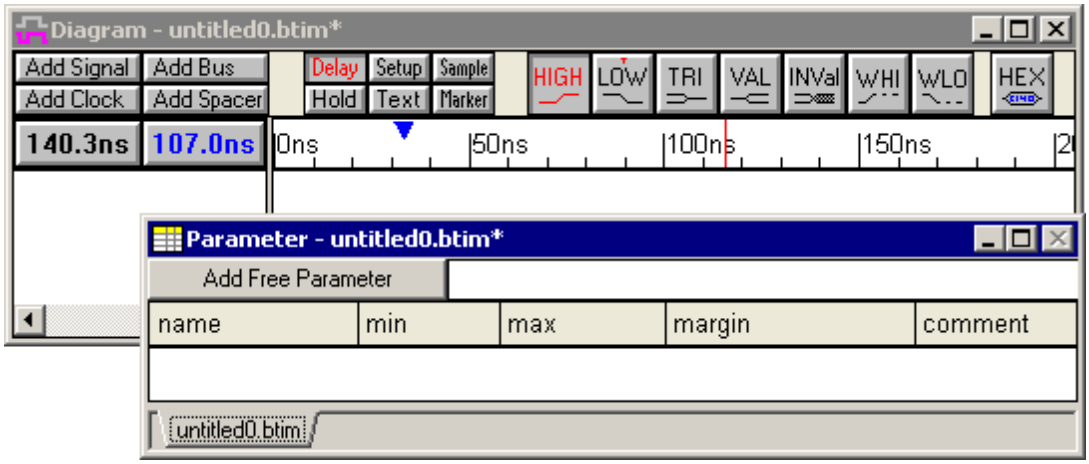
- Run one of the timing diagram editors from the Start Menu. This tutorial cannot be done with BugHunter Pro unless you also have a license for the Timing Analysis Option.

Copyright © 2014, SynaptiCAD Sales, Inc.



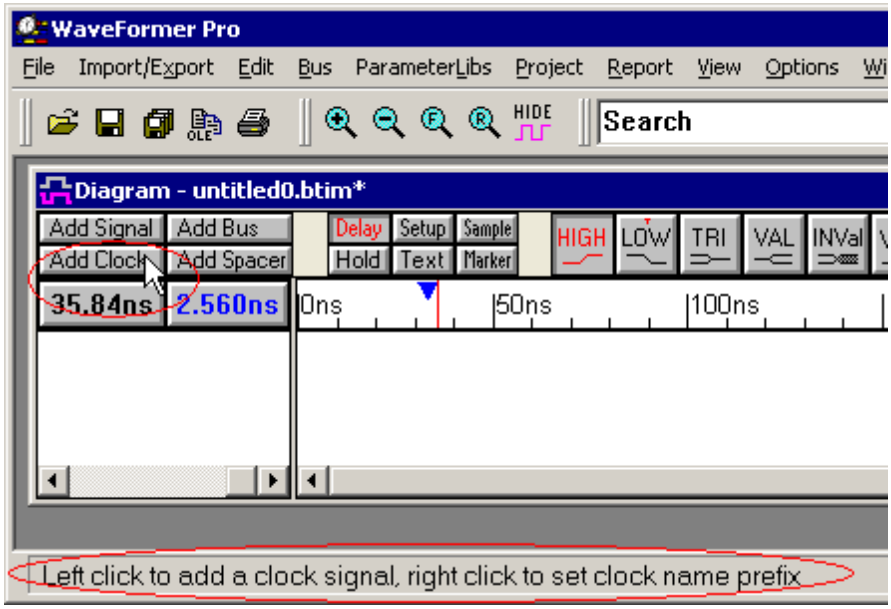
Open a New Timing Diagram File:

- Select the **File > New Timing Diagram** menu to open a diagram editing window and a matching parameter window.



Investigate the Status Bar:

- Move the cursor over the buttons on the diagram editor window, and then look at the bottom left corner of the big window to see the status bar.



- As you perform this tutorial, keep an eye on the status bar. It will give you hints on the types of

Copyright © 2014, SynaptiCAD Sales, Inc.

functions that the mouse can perform at that particular place. The status bar also works when the mouse is inside the drawing window.

(TD) 1.2 Set the Base and Dispaly Time Unit

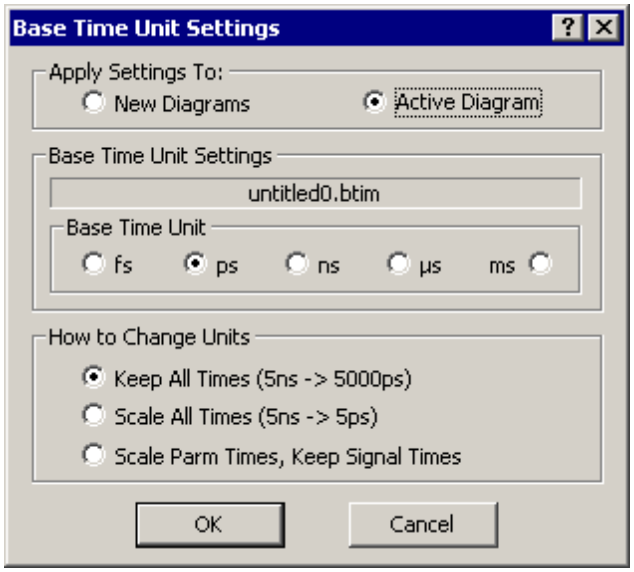
The **base time unit** controls the accuracy of the of time calculations. It is the smallest representable amount of time, and all time values are internally stored in terms of the base time unit. The **display time unit** controls the units for entering and displaying the results. At the beginning of a new design you should check these settings to make sure they are valid for the time ranges that you are working in.

Set the Base Time Unit:

The base time unit for your project should be set at least one unit below the units you are working in for best rounding performance during division operations (clock frequencies are inverted and stored internally as clock periods). The circuit that we are modeling in this tutorial has gate propagation times in the range of 3 to 18 nanoseconds and a clock with a period of 20ns. Therefore we will set the base time units to picoseconds.

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

- Check the **Active Diagram** box so that the dialog operates on your new timing diagram. Notice that the dialog displays the name of the diagram so that you can tell which diagram is the active one in a multi-diagram display.
- Check the **ps** button to set the base time unit to picoseconds.
- The remaining options control how any existing parameters or signals are changed when the base time unit is changed and have no effect on an empty timing diagram.
- Press the **OK** button to close the dialog.

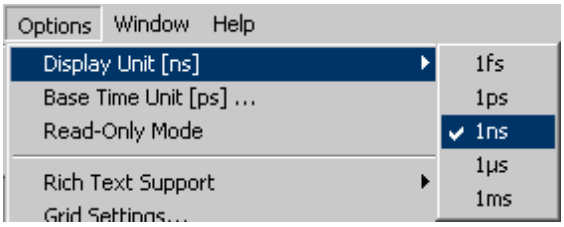


Set the Display Time Unit:

Set the display time unit to the units you most commonly use in the design. In this tutorial, all of the times are listed in nanoseconds so that will be the easiest time setting to enter the values.

Copyright © 2014, SynaptiCAD Sales, Inc.

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

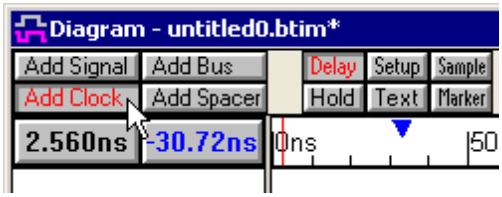


(TD) 1.3. Add the Clock

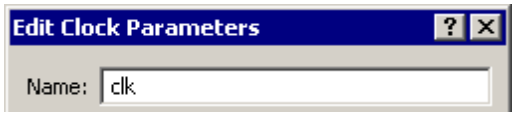
First we will create a clock. Clocks draw themselves based on their parameters, so you will not be able to drag and drop clock edges or make a delay end on a clock edge. For this tutorial, the clock is named **clk**, has a period of **50**ns (20MHz), and starts with a low segment.

Define the Clock Parameters using a dialog:

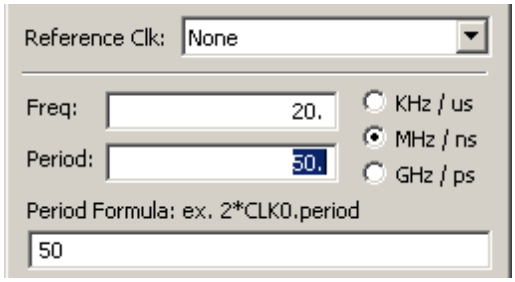
- Press the **Add Clock** button to open the *Edit Clock Parameters* dialog.



- In the **Name** box, type **clk** to set the clock name.



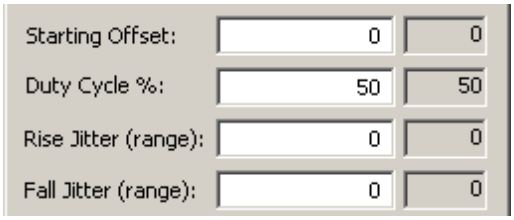
- In the **Period** box enter **50** and make sure that the **MHz/ns** button is selected. When you tab out of this box, the **Freq** box will change to 20 to match the new period value.
- Notice that the period can be also be defined by a **period formula** or in terms of a **reference clock**.



- Check the **Invert (Starts Low)** box at the bottom of the dialog. Clocks are normally displayed high at time zero, so "invert" makes the clock start low at time zero.

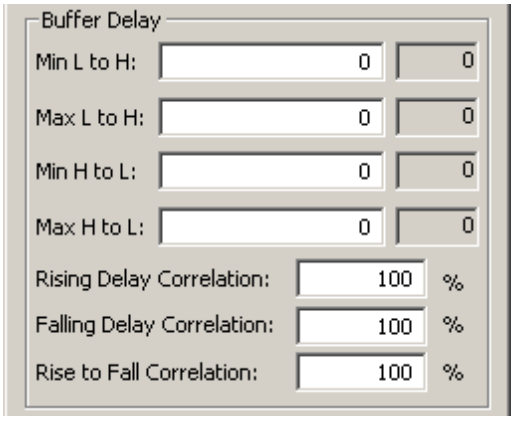


- Notice that the clock can have an offset starting time from time zero. The duty cycle can be set to any percentage value. And edge jitter is uncertainty around the occurrence of the clock edge.

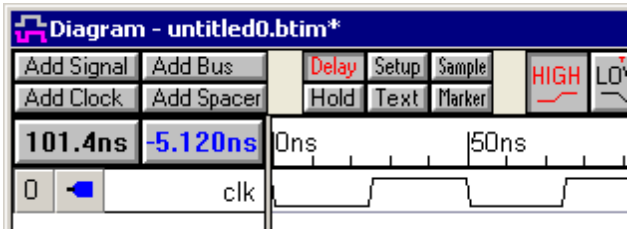


Copyright © 2014, SynaptiCAD Sales, Inc.

- Notice that buffer delays represent uncertainty after the clock edge (used to model uncertainty from clock tree buffers), and delay correlation determines how closely delays are related to each other.
- For more information on correlation and the different types of delays, and clock grids read Chapter 2: Clocks in the on-line help

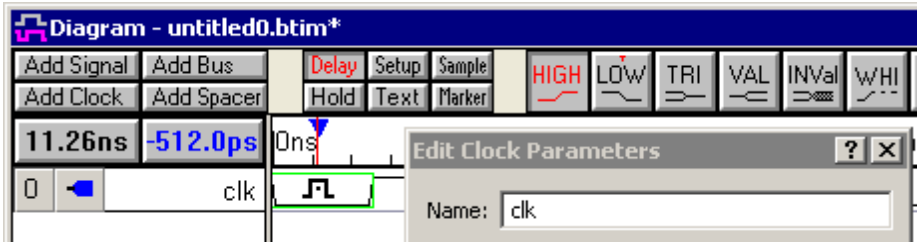


- Press the **OK** button to close the dialog. Make sure that the clock looks like the following image.



Reopen the Edit Clock Parameters dialog:

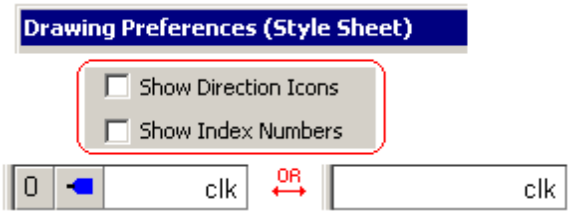
- Double left click on the clock waveform to reopen the *Edit clock parameters* dialog. Note, if you click too close to a clock edge it opens an edge dialog instead of the parameters dialog.



- Press the **Ok** button to close the dialog.

Hide the direction and index columns:

- The direction and index columns are not used in this tutorial so hide them by choosing **Options > Drawing Preferences** to open the dialog. Then uncheck **Show Direction Icons** and **Show Index Numbers**.



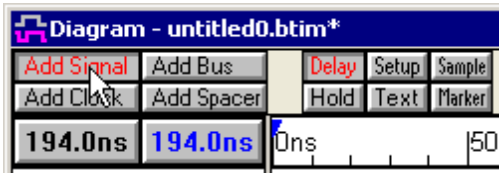
Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 1.4 Add the Signals

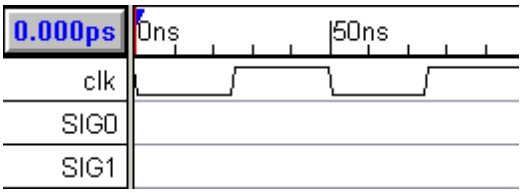
Next, add two signals and name them "Qoutput" and "Dinput".

Use the Add Signal button to create new signals:

- Press the **Add Signal** button two times to add two signals to the diagram window.

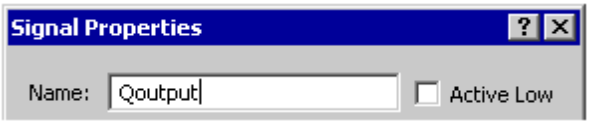


- The signals will have default names such as SIG0 and SIG1.

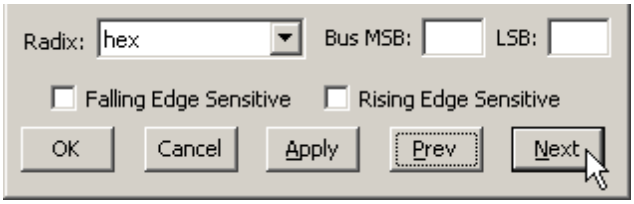


Double Click to rename the signals:

- **Double click** on the **SIG0** signal name to open the *Signal Properties* dialog.
- Enter **Qoutput** into the **Name** edit box. (DO NOT CLOSE THE DIALOG)



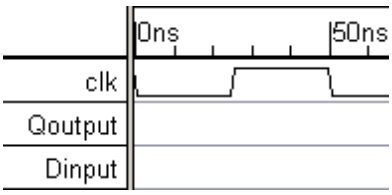
- Click the **Next** button or **ALT-N** to move to the next signal on the list. Notice that SIG1 is now displayed in the Name edit box.



- Enter **Dinput** into the *Name* edit box and press the **OK** button to close the dialog.



- The timing diagram should look like the following.



Tip: The *Signal Properties* dialog is a modeless dialog - you can keep the dialog open while working with other drawing features. The Boolean Equation and Simulation features of the *Signal Properties* dialog are covered in the Simulated Signals tutorial 36.

(TD) 1.5 Drawing Signal Waveforms

Next, we will draw some random waveforms to become familiar with the drawing environment. The timing diagram editor is always in drawing mode so left clicking on a signal will draw a waveform. The

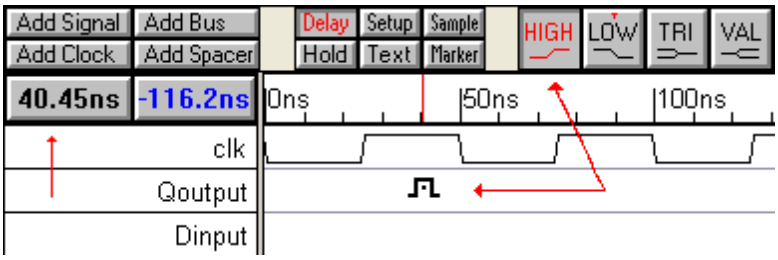
Copyright © 2014, SynaptiCAD Sales, Inc.

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.

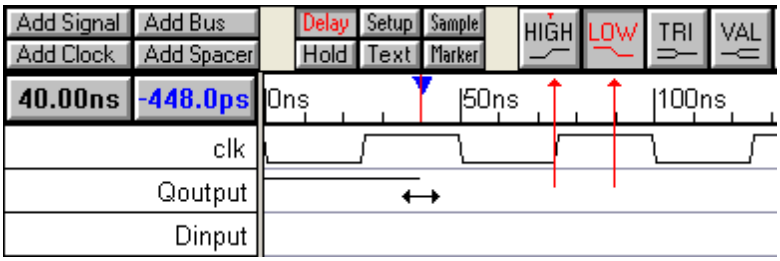


Draw and watch the State Buttons:

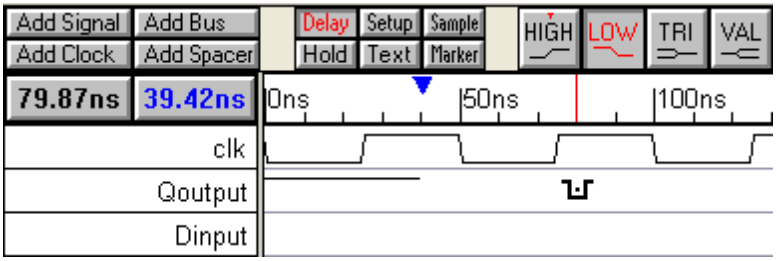
- Move the mouse cursor to about **40ns** and on the same level as **Qoutput**. Notice that the cursor has the same shape as the selected State Button.



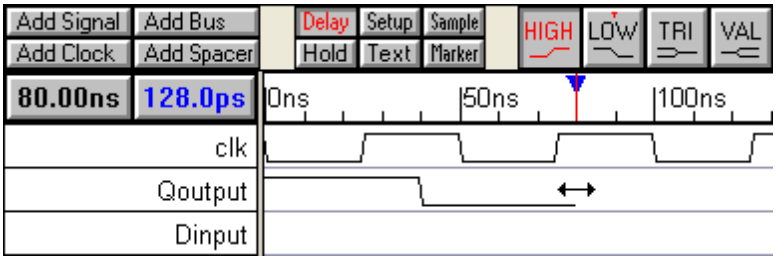
- **Left Click** to draw a high waveform segment from 0ns to the cursor. Notice that the State Button toggled to low, and the toggle T moved to the High button.



- Move the cursor to about **80ns** on the same signal. Notice that the cursor looks like a low signal to match the active state button.

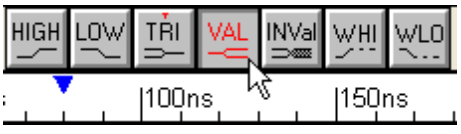


- **Left click** to draw a LOW segment. It is drawn from the end of the HIGH signal to the location of the cursor



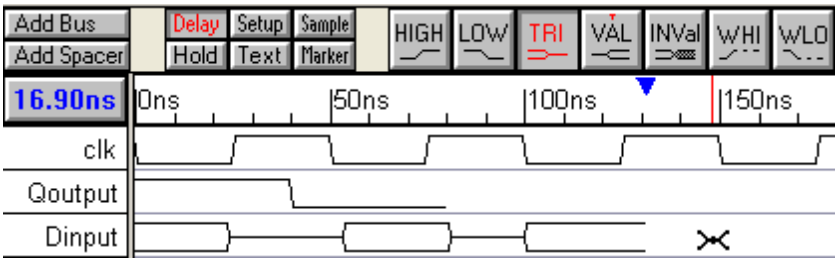
Draw with other state buttons:

- Left click first on the **Tri-state** button then on the **Valid** button, so that the Valid button is red and the tri-state button has the red toggle T on it.

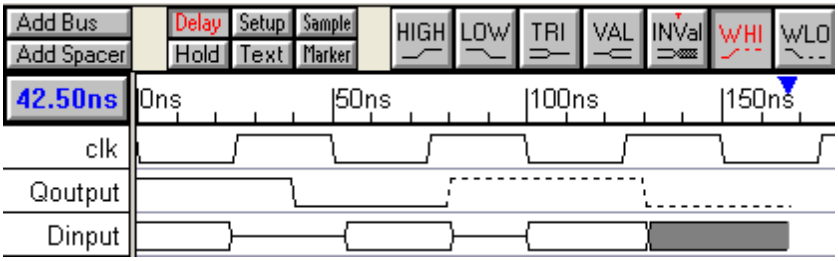


Copyright © 2014, SynaptiCAD Sales, Inc.

- Draw some valid and tri-state waveforms, while watching the cursor shape and the state buttons.



- Draw more segments, using all the states except the HEX button. The HEX state button is used in defining multi-bit signals and signals which have a user defined VHDL type. This button is covered in later tutorials. For now, experiment with the graphical states.



(TD) 1.6 Editing Signal Waveforms

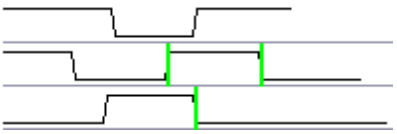
This section covers the main editing techniques used to modify existing signals (Note: these techniques will not work on clocks, because they draw themselves). The most commonly used technique is the dragging of signal transitions to adjust their location. Most of the other techniques all act on signal segments, the waveforms between any two consecutive signal transitions. The segment waveform can be changed, deleted, or a new segment can be inserted within another segment. Use each of the following techniques:

1) Drag-and-Drop Signal Transitions:

- Left click and hold down the mouse button on a signal transition and drag it to the desired location.

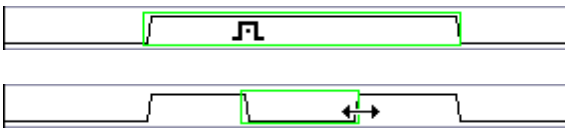


- To move transitions on different signals simultaneously, first select multiple transitions by holding the **<CTRL>** while clicking on edges. Then drag a transition to desired location.



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

- Inside of a segment, click and drag the cursor to insert a segment
- The inserted state is determined by the red state button

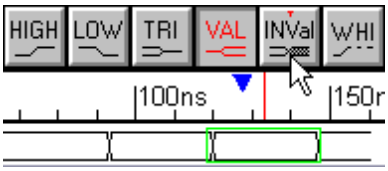


- **Delete a segment:** Select a segment (see above) and then press the **delete** key on the keyboard.

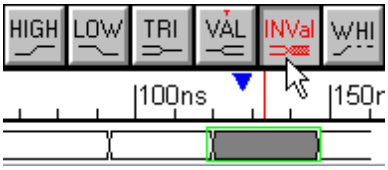
Copyright © 2014, SynaptiCAD Sales, Inc.

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).

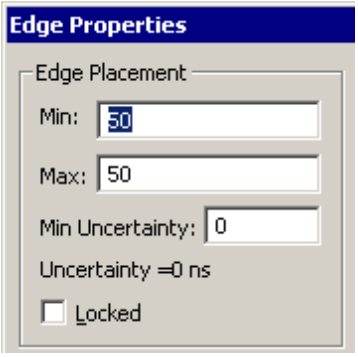


- Click on a state button to apply that graphical state to the segment. If you change a segment to the same state as an adjacent section, the transition will turn red to preserve the edge data. This transition can be deleted if necessary.



4) Find the exact edge time and see how to lock an edge

- Double-click on an edge of the signal transition to open the *Edge Properties* dialog.
- To move an edge, enter a new **min** or **max** time. An edge only has one time until uncertainty is added either by using a delay parameter or the **min uncertainty** box in this dialog.
- To lock an edge so that it cannot be moved, check the **Locked** checkbox. If a delay ends on a locked edge it will turn red if it cannot force the edge to the proper time.



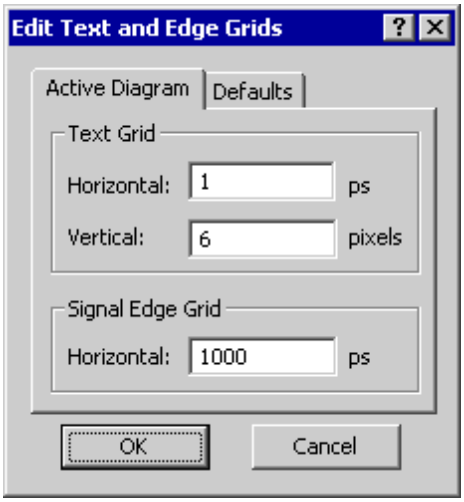
- Note: All edges on a signal can be locked by selecting the signal name, and then choosing the **Edit > (Un)Lock Edges of Selected Signals** from the main menu.
- Make sure to unlock any signals or edges you locked in this tutorial, or else your delay in the next section may not be able to force its ending edge.

5) Adjusting the drawing Grid

Drawn signal 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 display time unit, because this generates nice VHDL and Verilog stimulus generation files with whole number times (like 2ns instead of 2.465ns). However, it is sometimes convenient to set the grid to a multiple of the clock frequency to make all new signal edges line up with the clock edges.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **Options > Grid Settings** menu item to open the *Edit Text and Edge Grids* dialog.
- You do not have to make any changes to this dialog. Just notice that you are able to control the **Signal Edge Grid**.
- Also notice that text objects have a different grid.



6) Adding virtual state Information to a segment

- 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.



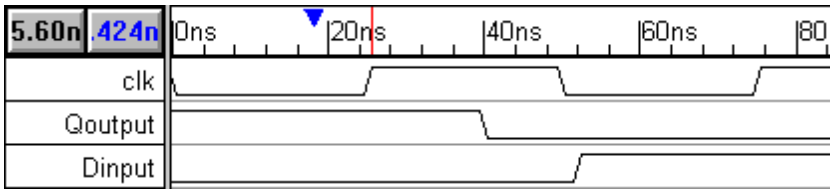
- For Clocks, press the **Hex** button and then double-click on the middle of the segment to open the *Edit Bust State* dialog. If the Hex button is not pressed, the double-click will open a different dialog to allow editing of the clock.

(TD) 1.7 Adjust Diagram to Match Figure

In the next few sections we will be adding the delays and setups to the timing diagram. It is best to start with the waveform edges in the approximate position that they should be in when the timing diagram is finished.

Adjust the Waveforms:

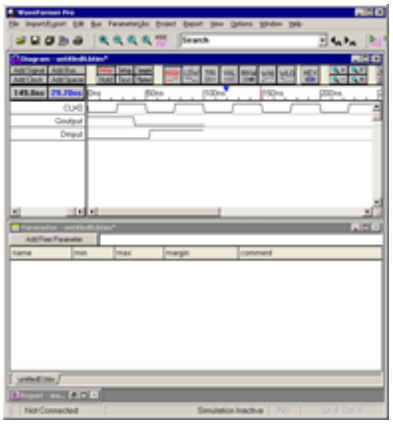
- Use the editing techniques in the previous section so that the waveforms have roughly the same transitions as the signals in the figure below.



Copyright © 2014, SynaptiCAD Sales, Inc.

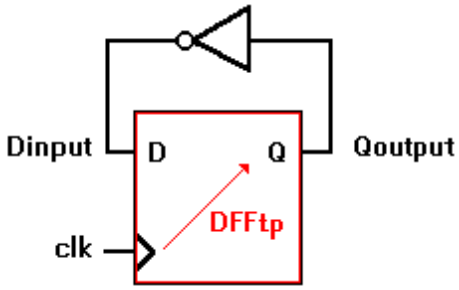
Minimize the Report Window and tile the Parameter and Diagram Windows:

- Minimize the *Report* window because it is not used in this tutorial.
- Select the **Window > Tile Horizontal** menu to tile the *Parameter* and *Diagram* windows so that you will be able to see the interaction between the two windows.



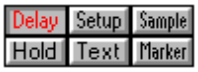
(TD) 1.8 Add the D Flip-Flop Propagation Delay

Add the delay that represents the propagation time from the positive edge of the clock to the Qoutput of the D flip-flop. First we will add a blank delay between the edges and then we will edit it so that the delay is named "DFFtp" and has a propagation delay of 5-18ns.



Add a blank graphical delay:

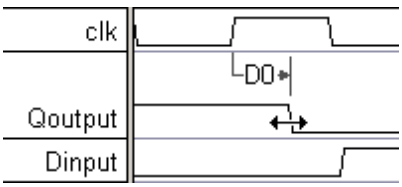
- Press the **Delay** button so that right clicks will add delays.



- Left click on the first rising edge of the clock to select it. This edge will be the first or driving edge of the delay.

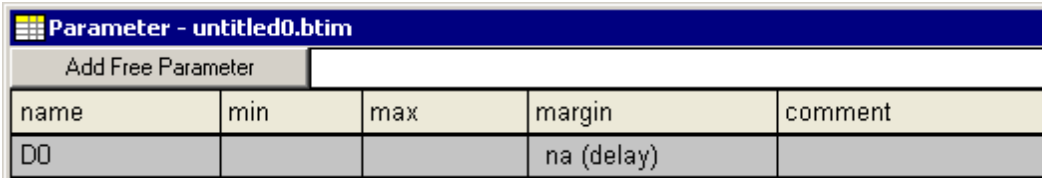


- Right-click on the first falling edge of the **Qoutput** signal, to add the delay between the two edges. Since the delay is pointing to this edge, this will be the edge that moves in response to formula values entered into the delay



- Notice that D0 was also added to the *Parameter* window.

Copyright © 2014, SynaptiCAD Sales, Inc.

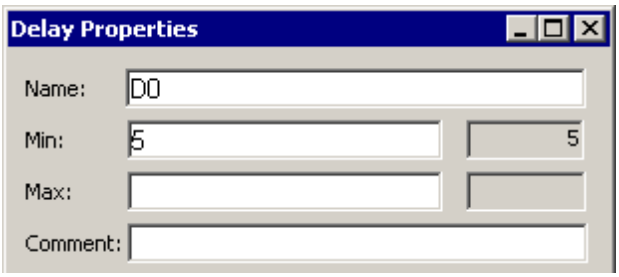


Watch the delay as you change the min value:

When delays are added, they are blank and do not enforce any timing restraints. Notice that the delay is drawn with gray colored lines: this indicates that the delay is not forcing either the min or max edge of the **Qoutput** signal. Now edit the delay's parameters.

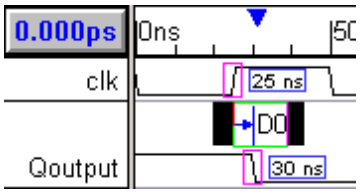
- **Double-click** on **D0** in either the Diagram or Parameter window to open the *Delay Properties* dialog. For simplicity, we will refer to this dialog as ***Parameter Properties***, even though the name at the top may say *Delay Properties* or *Setup Properties* depending on the type of parameter being edited.

- Adjust the position of the *Parameter Properties* dialog so that you can see the parameter in the *Diagram* window and at least part of the parameter in the *Parameter* window.
- Type **5** into the **min** edit box and press the **TAB** key to move to the **max** edit box (leave max blank for now). This enters 5 display time units (5ns for this timing diagram).



Several things happened when you pressed the TAB key:

- The falling edge of **Qoutput** was moved so that it is 5ns from the clock edge. If you hover over the delay with the mouse, blue boxes with the edge times will appear so that you can check the edge times.



- Also note that the delay changed from a gray color to a **blue** color. Delays are color-coded to indicate which delays are forcing the min and max edges of a transition. This type of critical path display is necessary in diagrams where multiple delays drive a single signal transition. The colors are: Gray = none, Blue = Min only, Green = Max only, Black = both min and max. After this tutorial you may want to experiment with the **multdely.btim** file (in the Examples directory) to see the effects of multiple delays on a single transition and critical path color coding.

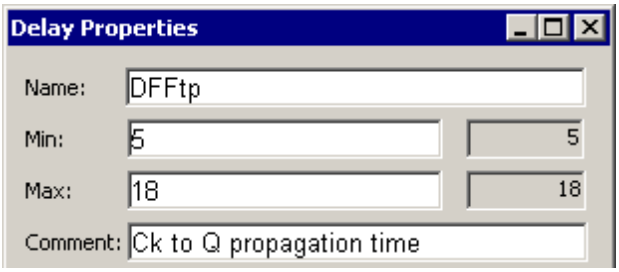
- Finally, the parameter information also was updated in the *Parameter Window.*

name	min	max
D0	5	

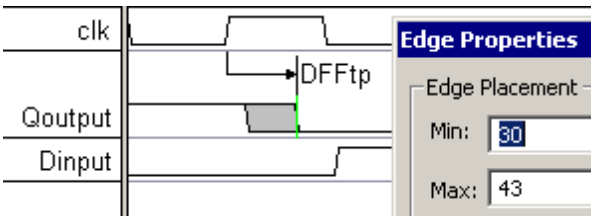
Copyright © 2014, SynaptiCAD Sales, Inc.

Edit the rest of the delay:

- Type **DFFtp** in the **Name** box.
- Type **18** in the **Max** box.
- In the **Comment** box, enter **Ck to Q propagation time**.
- Close the dialog when you are done.



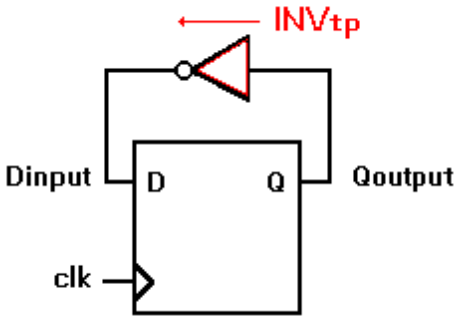
- Notice that the DFFtp delay is black which indicates that it is forcing both edges of **Qoutput**.
- Also notice the falling edge of Qoutput now has a gray uncertainty region. Double click on the edge to verify that the edges of the region are **5ns** and **18ns** from the clock edge (13ns of uncertainty).



Tip: The *Parameter Properties* dialog is **modeless** (other operations can be performed while the dialog is open) and **interactive** (any changes in the dialog fields are reflected in the diagram after you move out of that field). When the *Parameter Properties* dialog is open you can edit a different parameter by double-clicking in the *Diagram* or *Parameter* window on the parameter you want to change. If you double-click in the *Diagram* window, that instance of the parameter will be edited (the **Change All Instances** checkbox will NOT be checked). If you double click in the *Parameter* window, ALL instances of the parameter will be edited (the **Change All Instances** checkbox will be checked).

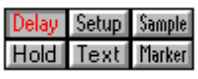
(TD) 1.9 Add the Inverter Propagation Delay

Next add the delay that represents the propagation time of the inverter from its input Q to its output D. Since this delay is the second in a chain starting with DFFtp, its uncertainty region will be larger than just the uncertainty caused by the inverter.



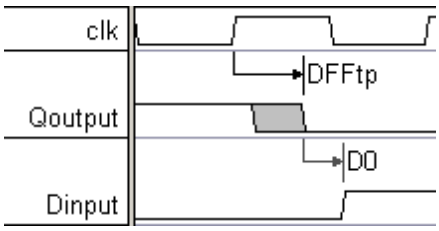
Add the Inverter Delay:

- Make sure the **Delay** button is red so that right clicks will add delays.

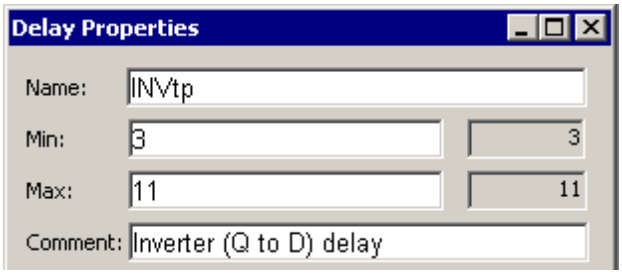


Copyright © 2014, SynaptiCAD Sales, Inc.

- Left click on the first falling edge of the **Qoutput** signal to select it (the same edge that ends the "DFFtp" delay).
- Right-click on on the first rising edge of the **Dinput** signal to add a blank delay.



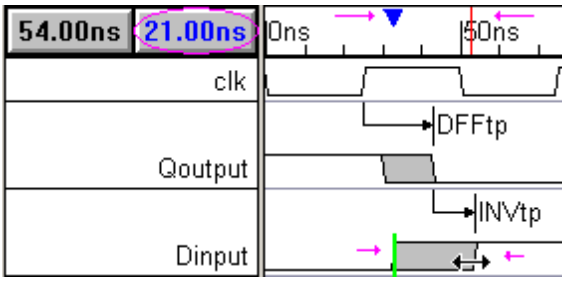
- Double-click on the new delay to open the *Parameter Properties* dialog and enter the following values: Name is **INVtp**, propagation delay of **3** to **11** ns, and a comment of **Inverter (Q to D) delay**.
- Click on the **OK** button to close the dialog.



Verify that the Uncertainty region is correctly calculated:

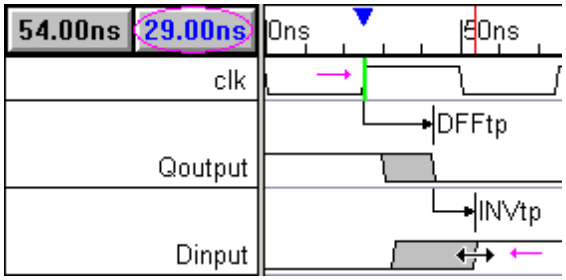
Notice the uncertainty region for the *Dinput* transition is much larger than the 3-11 ns that you entered in the last step. This is because the DFFtp uncertainty adds to the INVtp uncertainty.

- Click on the first rising edge of **Dinput** (to select it). This also moves the blue delta mark on the time line.
- Move the mouse cursor over the second edge of the uncertainty region. As you move the red line on the time line tracks your progress, and the Blue delta readout shows the exact distance from the blue delta mark.



- Here the readout shows that the uncertainty region lasts for 21ns (13ns from DFFtp + 8ns from INVtp = 21ns).

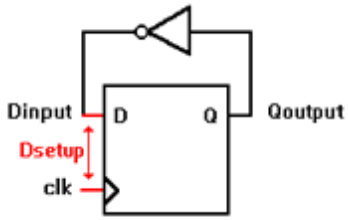
- Next, click on the first edge of *clk* and measure to the end of the uncertainty region of *Dinput*. If both the inverter and the D flip-flop are slow, *Dinput* may not transition until 29ns after the clock edge.



Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 1.10 Add the Setup for the Dinput to Clock

One of the most important features of a timing diagram editor is that setup and hold parameters can monitor pairs of signals transitions to make sure that they do not violate the timing constraints of the circuit. In this design, if Dinput changes too close to a clock edge then there is a risk that the flip-flop will go into a meta-stable state. We will use a setup parameter to make sure the Dinput does not violate the setup time for the clock.

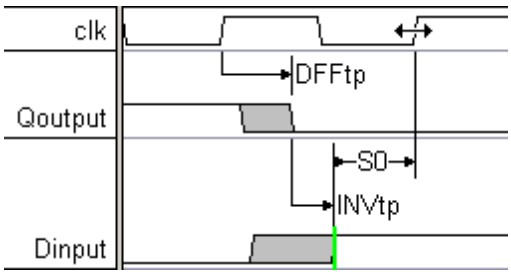


Add a Setup parameter:

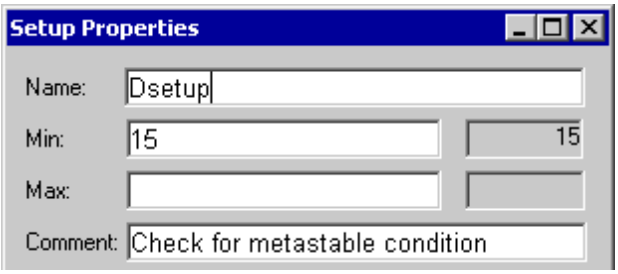
- Press the **Setup** button so that right clicks will add setups.



- Left click on one of the rising edges of the **Dinput** signal to select it.
- Right click on the second rising edge of the **clock** to add a blank setup between the selected edge and this one.
- Notice that the arrows of the setup are pointing to the control signal. This means that you added the setup correctly.



- Double-click on the new setup to open the *Parameter Properties* dialog and enter the following values: Name is **Dsetup**, min time is **15**, and the comment is **Check for metastable condition**.
- Press the **OK** button to close the dialog.



Notice that the margin column in the *Parameter* window says that there is a **6ns** safety region before the setup is violated. Verify this by clicking on the maximum edge of the **Dinput** signal (to place the blue delta mark on the time line), then placing the cursor on top of the second rising edge of the clock. The blue time readout should say 21ns (measured time 21ns - setup time 15ns = 6ns margin).

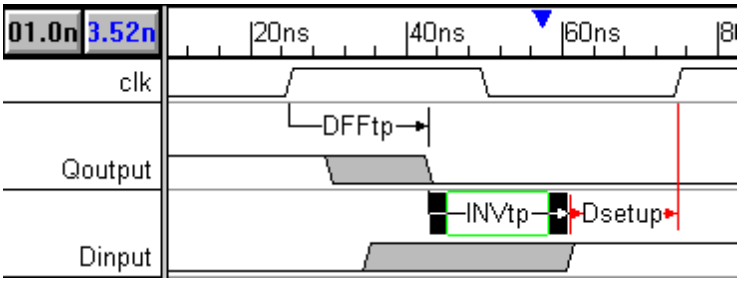
name	min	max	margin	comment
DFFtp	5	18	na (delay)	Ck to Q propaga
INVtp	3	11	na (delay)	Inverter (Q to D)
Dsetup	15		6	Check for metas

Cause the Setup to be violated:

Next, we will demonstrate what happens when a setup is violated by increasing the inverter's delay.

- Double-click on **INVtp** to open the *Parameter Properties* dialog and change the **max** time to **18** ns. Then press the **Apply** button to apply the change.

Copyright © 2014, SynaptiCAD Sales, Inc.



- Notice that the setup has turned red in the *Diagram* window and that the **Margin** value of the *Parameter* window has also turned red.
- Change the inverter delay back to **11ns** and click **OK** to close the dialog.

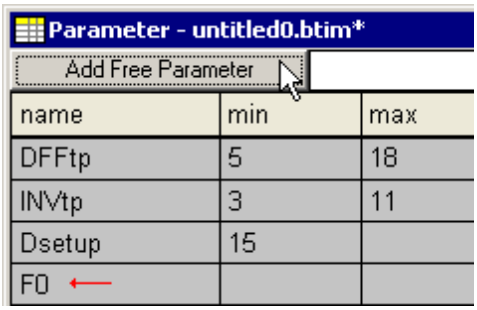
(TD) 1.11 Add a Free Parameter

So far we have always directly edited a parameter's values. This is inefficient and error prone if the circuit is large. It would be better to define one variable that held the value and make everything that needed that value reference this variable. Then if the value needs to be changed, you only have to edit one variable.

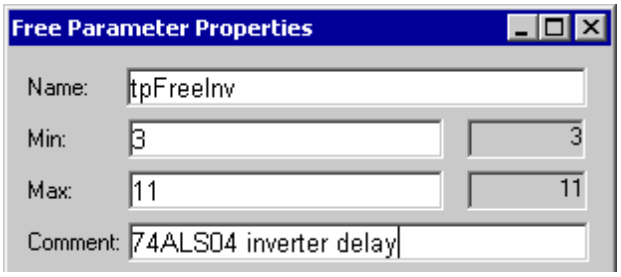
Free parameters act as variables that can be referenced by other parameters. They are called "free" because these parameters are not attached to any signal transitions in the *Diagram* window. Let's add a free parameter to hold the propagation times for the inverter.

Add the free parameter:

- In the *Parameter* window, press the **Add Free Parameter** button to create a blank free parameter.

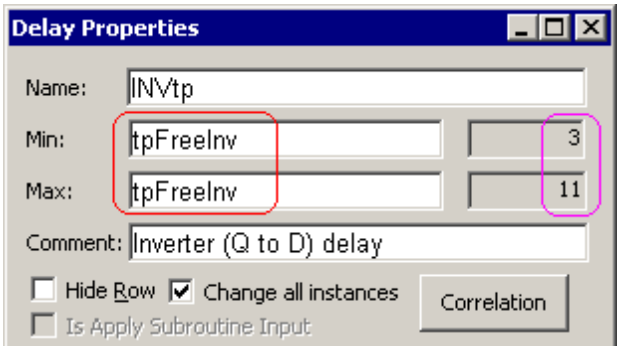


- Double click on the free parameter to open the *Parameter Properties* dialog box and enter the following: Name is **tpFreeInv**, min time is **3**ns, max time is **11**ns, and comment is **74ALS04 inverter delay**. (Leave the dialog open).

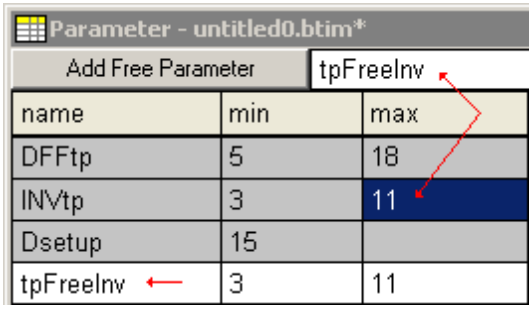


Copyright © 2014, SynaptiCAD Sales, Inc.

- Make the dialog point to the **INVtp** delay, either by pressing the **Previous** button, or by double clicking on **INVtp**.
- Type **tpFreeInv** into the min and max cells of **INVtp** and notice that the calculated values show the actual times. Any changes to the timing values of the free parameter will now affect this delay.



- Notice that the row for **tpFreeInv** turned white to indicate that it is being referenced by another parameter.
- Select the max value of INVtp in the parameter window and notice that the formula is displayed in the box above.



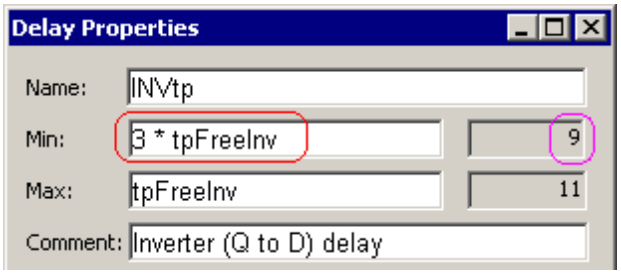
- You can make the parameter window display formulas in the table part by choosing the **Options > Parameter Window Preferences> Display min/max formula**.



Using Formulas in the Parameter time boxes:

Parameters can contain mathematical formulas as well as numeric time values. Common operations include multiplication(*), division(/), addition(+), and subtraction(-). For more information in the syntax for formulas see the Timing Diagram Editor Manual Section 2.5 Time Formulas for Clocks and Parameters. For example, the inverter in this circuit could represent 3 cascaded inverters used to generate a minimum delay of 9ns. To represent this in your timing diagram:

- Enter **3 * tbFreeInv** into **INVtp**'s **min** edit box. Then tab to a new box and see that the equation correctly calculated **9ns**.



- Free parameter names can also be used with an attributed parameter name such as **tpFreeInv.min** and **tpFreeInv.max**. This gives you the flexibility to specify formulas any way you need. If no attribute is added then a min or max is assumed depending on whether the formula is in the min or max column.

Create Libraries of Free Parameters:

Free parameters can be saved to special library files which can later be merged into other projects.

Copyright © 2014, SynaptiCAD Sales, Inc.

You can also reference free parameters without including them into your project file by placing libraries in your library search path (**Libraries > Library Preferences** menu option). For more information on free parameters and libraries read the on-line help *Chapter 10: Libraries* or perform the Parameter Libraries Tutorial 90.

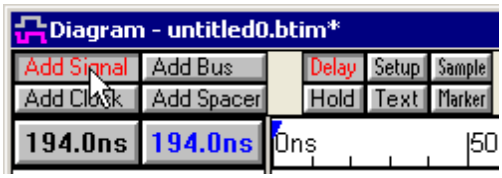
(TD) 1.12 Drawing with Equations

We have finished with the timing analysis section of this tutorial, and next we would like to take the time to show you a few more drawing techniques that will help you create and manage complex timing diagrams. One such technique is to use equations to draw and label waveforms. Waveform and label equations provide a quick way to generate signals that have a known pattern that is more complicated than a periodic clock. WaveFormer (and higher editors) also support simulated signals based on Boolean Equations which are covered in the Simulated Signal tutorial 36.

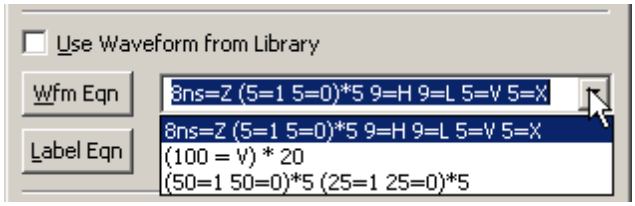
Use the Waveform Equation Feature:

The waveform equation box in the *Signal Properties* dialog accepts a list of time/value pairs, and the default equation has all of the syntax and states that are supported by this box. Each time you press the button more waveforms will be added on to the end of the signal.

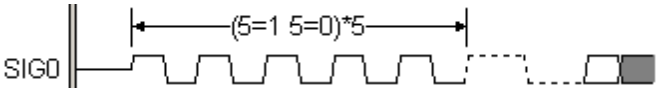
- Press the **Add Signal** button two times to add two new signals to the diagram window.
- Double click on **SIG0** to open the *Signal Properties* dialog.



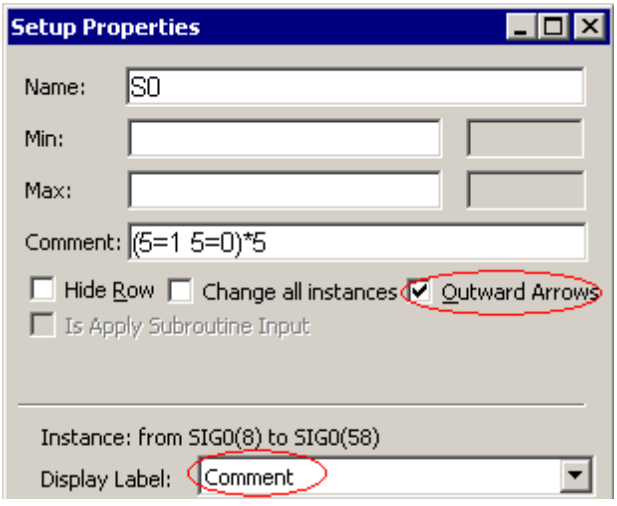
- Press the **Wfm Eqn** button to apply the default equation to SIG0. This equation shows the syntax for all possible waveform types. Look at the waveform and try to match it to the different parts of the equation.



- The first pair, **8ns=Z**, causes an 8ns long tri-state segment to be drawn.



- The next terms, **5=1 5=0**, draws a 5 ns long high segment followed by a 5ns low segment, where the **ns** is implied by the display time unit. Enclosing it in **(...)*5** causes the sequence to be repeated 5 times.
- The other pairs are interpreted in a similar manner. The values H and L draw weak high and low waveforms, and V and X draw valid and invalid sections.
- We annotated the last sequence using a setup parameter and changing the display label from name to comment.



Copyright © 2014, SynaptiCAD Sales, Inc.

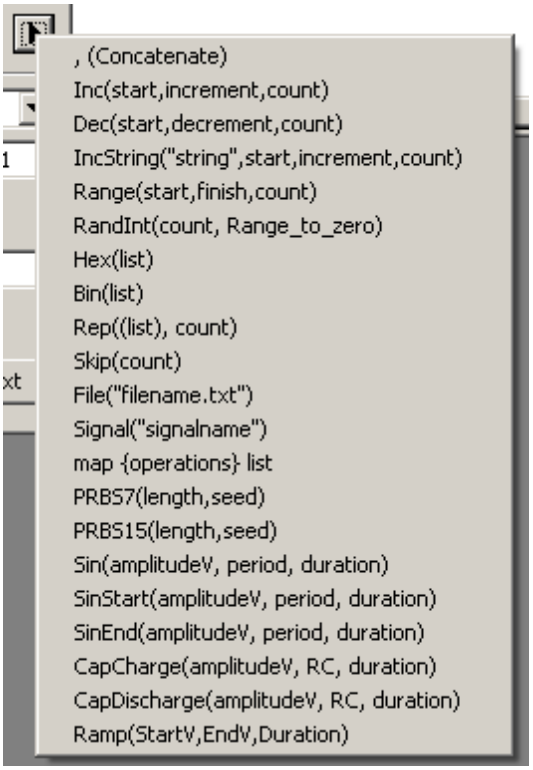
Use the Label Eqn Feature:

Label equations are used to automatically insert data on waveform segments. All of the equations are listed in the label fly-out box. Here we will draw and label a counter signal that first counts up and then counts down.

- Double click on **SIG1** to open the *Signals Properties* dialog, then use a waveform equation to draw seven valid segments that are 10ns long.



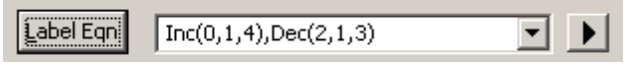
- Open the fly-out to the right of the **Label Eqn** button and take a look at the list of available functions. Choosing any function puts it at the end of the current label equation, then you can edit the parameters of the function call.
- Functions can be concatenated together by separating them with a comma.
- The Analog Label equations for sine waves, capacitor functions, and ramps are covered in the Analog Signals Tutorial 70.



- Use the Label equation quick-fill box and choose **Inc** then **Concatenate** and then **Dec** to add those equations to the edit box.

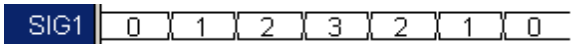


- Edit the parameters as shown, so that the counter first starts from 0, adds 1 each time, and counts up for 4 cycles. Then make the counter count down starting at 2, for 3 cycles.



Copyright © 2014, SynaptiCAD Sales, Inc.

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



(TD) 1.13 Drawing Virtual Busses

Buses are multi-bit signals. The timing diagram editor supports several different kinds of buses to accommodate all the different ways signal information may be imported or exported from the tool. Virtual and Group buses have the ability to be converted from one to the other type by right clicking on the name.

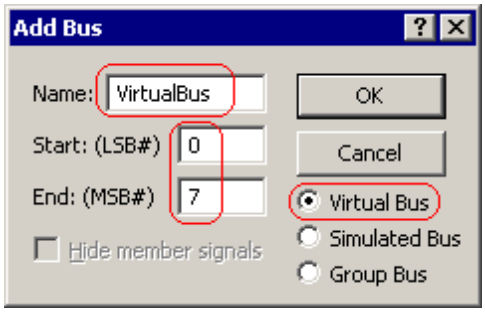
- **Virtual Bus** is a single signal defined as multiple bits. This is the most common and easiest to work with because all of the normal signal editing techniques work on it.
- **Group Bus** displays the aggregate values of its member signals. This is handy way to manage lots of single bit signals that have been imported from other sources (like logic analyzers).
- **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. This is covered in the Simulated Signals Tutorial 36.
- **Differential Signals** are two-bit group buses that display a superimposed image of the member signal waveforms. This can also be a useful technique for overlaying two analog signals to compare them visually.

Draw a Virtual Bus:

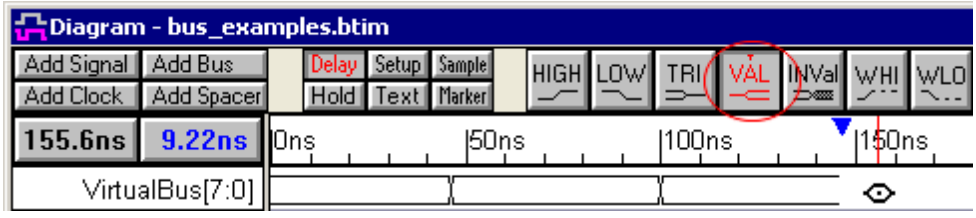
Virtual Buses are the recommended way to display and work with bus information. Virtual Buses are regular signals that have the LSB and MSB values set.

- Add an 8-bit virtual bus named **VirtualBus** with an LSB of 0 and an MSB of 7 using one of the following methods.

- **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.

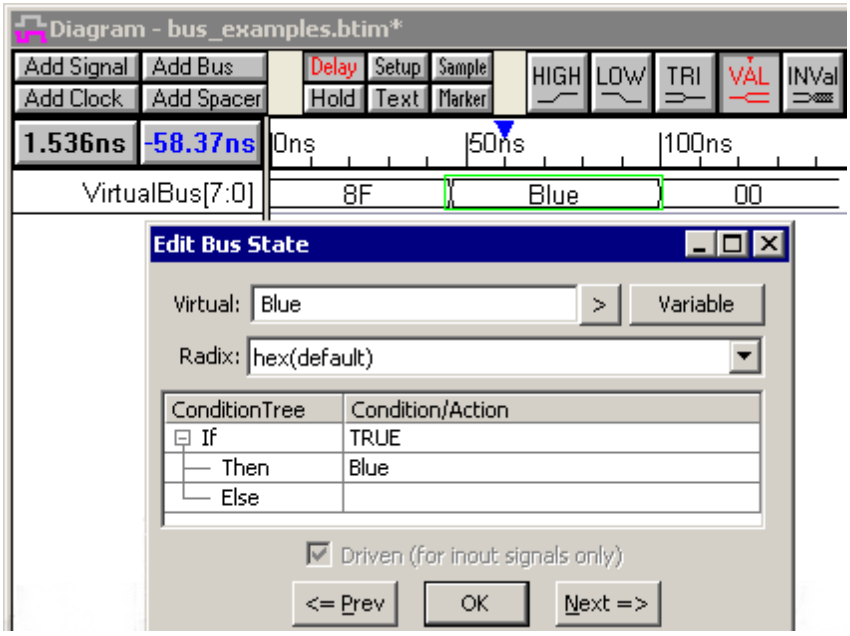


- You can sketch the virtual bus waveform using any of the graphical states, but normally virtual buses are drawn with all valid states. Press the **Valid** state button twice so that it is red and also has the red T on the top of the button. Then draw some consecutive valid states.



Copyright © 2014, SynaptiCAD Sales, Inc.

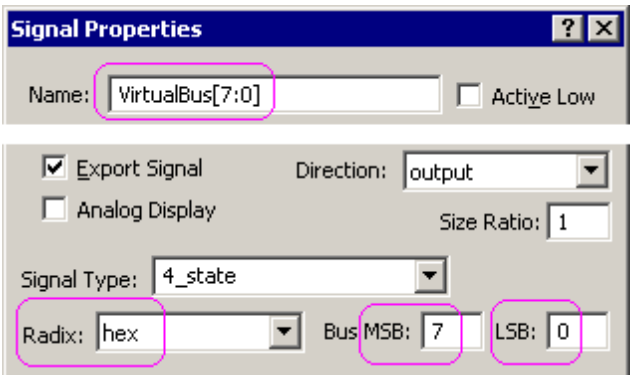
- Open the *Edit Bus State* dialog by either double-clicking on a segment OR first selecting a segment and then clicking the **HEX** button on the button bar.



- 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. When you are done press the **OK** button to close the dialog.

Investigate the Virtual bus using the Signal Properties dialog:

- Double click on the **VirtualBus** signal name to open the *Signal Properties* dialog.
- On the bottom of the dialog, notice the **MSB** and **LSB** settings are the same as what you typed in the *Add Bus* dialog
- Notice the **Radix** setting which controls how the tool interprets the data in the virtual states of the waveform.



(TD) 1.14 Drawing Group Buses and Differential Signals

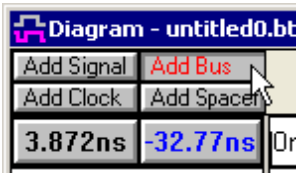
Group Buses display the aggregate value of their member signals. Normally, you would use them after importing from a format that treated all signals like one-bit signals (like from a logic analyzer). They are also used to create differential signals, which are just two bit group buses with some special display settings. Before a group bus can be created, its member signals must either be specified by selecting the signal names or new signals need to be created. We will use both methods in this

Copyright © 2014, SynaptiCAD Sales, Inc.

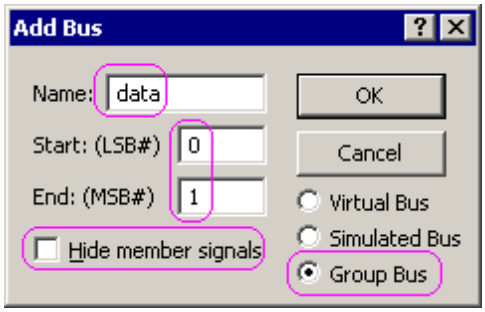
tutorial.

To create a group bus and its member signals:

- Make sure that no signal names are selected (clear selected signals by clicking in the *Diagram* window), then press the **Add Bus** button to open the *Add Bus* dialog. If signals are selected, they will become the member signals of the bus (we will do that next).

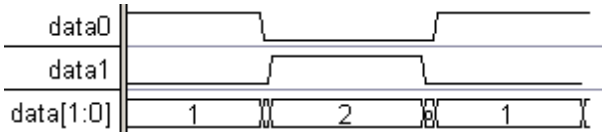


- Name the bus **data** and set the LSB to **0** and the MSB to **1**.
- Check the **Group Bus** button.
- Verify that the **Hide member signals** check box is **NOT** checked. We want to be able to see the member signals in this demonstration.

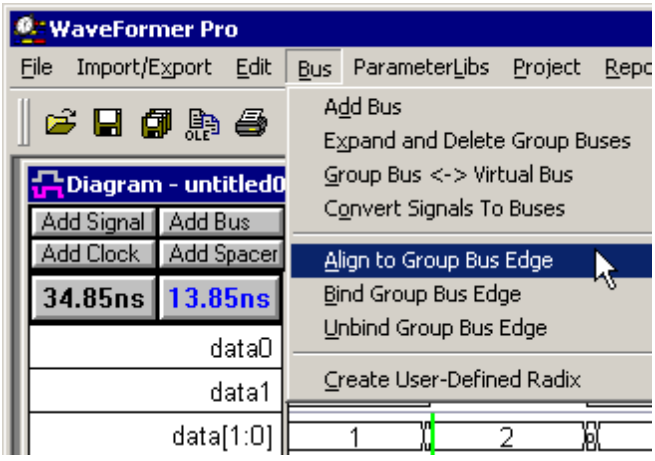


- Press the **OK** button to close the dialog and create the bus. There should be 3 signals generated: **data** (the bus), and **data0** and **data1** (the bus member signals). If the member signals are not shown, use the **View > Show Hidden Signals** to show them.

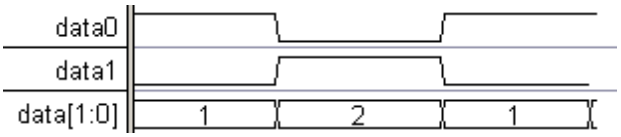
- Draw High and Low segments on **data0** then draw the opposite on **data1** (later we will use these to make a differential signal).



- Notice that the edges in the **data** bus are a little ragged because the edges of the member signals are not on exactly the same times.
- Select an edge on **data** and choose the **Bus > Align to Group bus edge** to snap all the edges together. All the edges can be locked together by using the **Bind** menu.

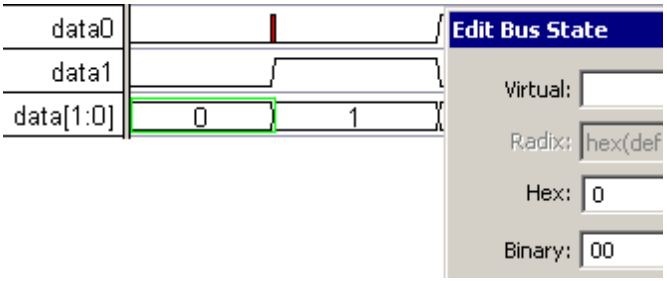


- Note the bus edge can be locked to the member edges at a particular time by selecting and edge and choosing the **Bind Group Bus Edge** menu as shown above.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Note that you can edit the **data** bus values by double clicking on a segment to open the *Edit Bus State* dialog and changing the **Hex** or **Binary** state (but not the virtual state). The member signals change to reflect the new value.

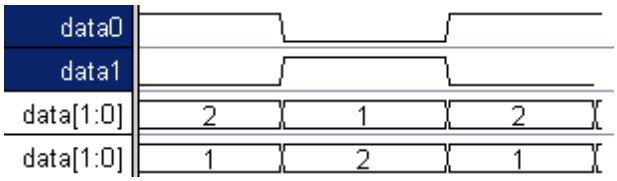


- The red event on **data0** preserves the edge on the member signal, so that you can make consecutive changes in the bus values without stopping to add edges. To remove the red events choose the **Edit > Clear Red Events** menu, but don't to it now. Just **return the first state to 1**.

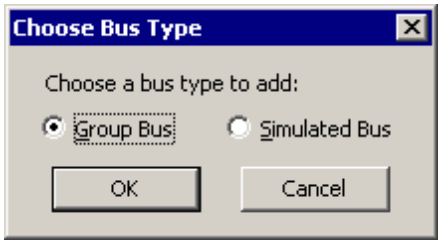
Creating a group bus from existing signals:

Here we will create a bus using existing signals by selecting the signal names in order from LSB to MSB, then adding the bus. We will make a bus with the opposite order from the last bus.

- Select **data1** by clicking on the name. This will be the LSB of the new bus.
- Select **data0** by clicking on the name. This will be the MSB of the new bus.



- Press the **Add Bus** button to open the *Choose Bus Type* dialog. Notice that the *New Bus* dialog did not open up because this bus will be automatically created from the selected signals.

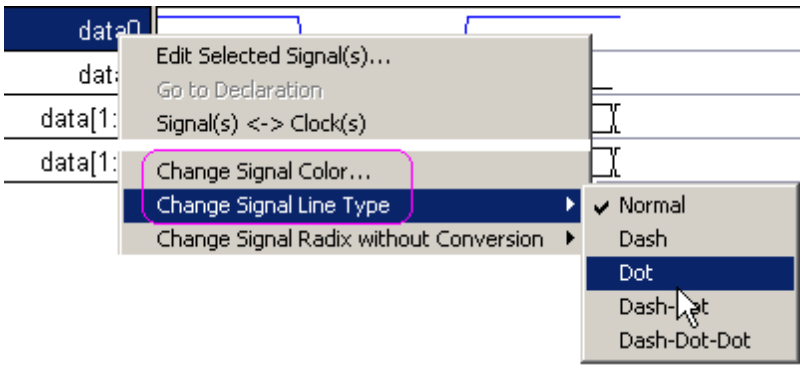


- Select the **Group Bus** radio button and click **OK** to close the dialog. Notice that a new bus, **data**, was added to the diagram and that it has a different **MSB** and **LSB** than **data**.

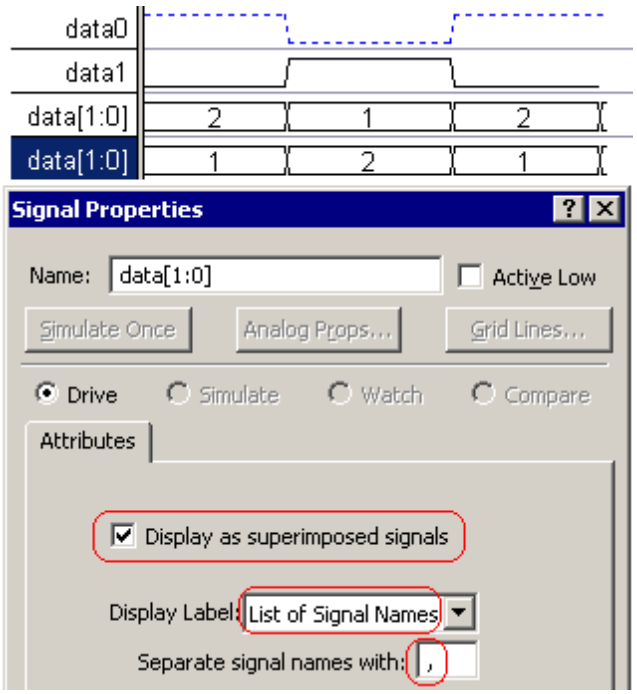
Create a Differential Signal

- Differential signals look best when the line type and color of one of the member signals is different. Right click on the **data0** signal and use the menus to change the *signal color* to **blue** and the *signal line type* to **dot**.

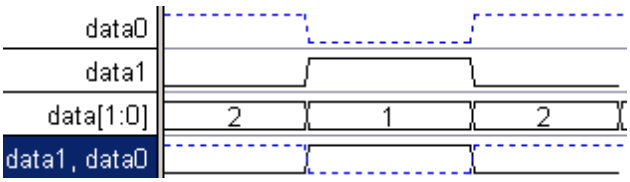
Copyright © 2014, SynaptiCAD Sales, Inc.



- Double click on the second **data** bus to open the *Signal Properties* dialog.
- Check the **Display as Superimposed signals** so that the signals will draw on top of each other, instead of the normal state value display.
- Change **Display Label** drop-down to **List of Signal Names** so that the bus name is replaced with the list.



- Press the **Ok** button to close the dialog and display the bus as a differential signal.

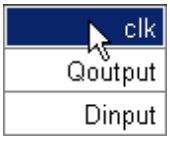


(TD) 1.15 Working with Drawing Environnment

Play with moving signals and zoom in and out of the diagram.

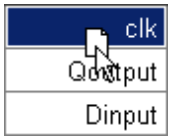
Move a single signal:

- Select the signal **clk** by clicking on the name (a selected signal will be highlighted).

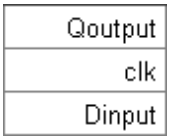


Copyright © 2014, SynaptiCAD Sales, Inc.

- Click down and hold on the selected signal so that a paper icon appears.



- **Drag** the paper icon until it is between **Qoutput** and **Dinput,** Then **Drop** the icon by releasing the mouse button. Notice the timing diagram has redrawn itself.



- Try dropping **clk** at the very top and at the very bottom of the diagram. Leave **clk** at the bottom of the diagram.

Moving and reordering multiple signals:

When several signals are highlighted and moved as a group, they will reorder themselves according to the order in which they are selected. This ability to quickly reorder signals by the order of selection will help you deal with the large numbers of member signals of buses.

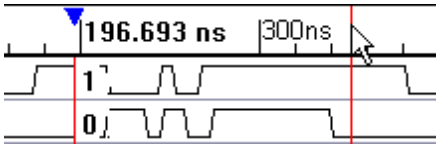
- Hold the CTRL key while first selecting **Dinput**, then selecting **Qoutput** by left clicking on the signal names in that order.
- **Move** the signals to the bottom of the diagram. Notice that Dinput is above Qoutput because that is the order in which they were selected.
- Select **Qoutput** and then select **Dinput** (don't forget to use the CTRL key).
- **Move** the signals to the top of the diagram. Notice that Qoutput is above Dinput, because the signals were selected in that order. This is a quick way to reorder a large group of signals.
- Return the signals to their original order, (clk, Qoutput, Dinput).

Play with the Zoom Level

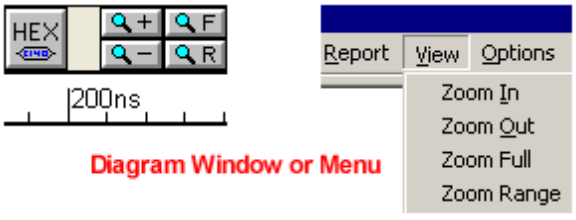
- To zoom in and out quickly, hold down the **<Shift>** key while using the **scroll wheel** on your mouse.

<Shift> and mouse **scroll wheel**

- To zoom in over a visible section, drag and drop inside the **Time Line.**



- The zoom buttons are located on menu bar in the diagram window button or 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.

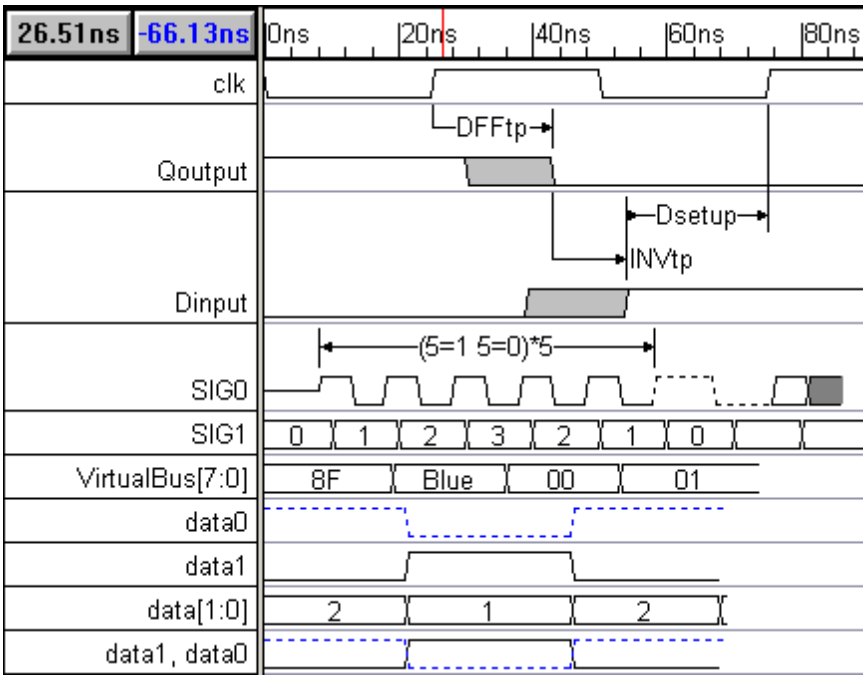


- The zoom full (F) displays the entire timing diagram on the screen.
- The zoom range (R) opens a dialog that lets you specify the starting and ending times for the zoom.

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 1.16 Summary

Congratulations! You have completed the **Basic Drawing and Timing Analysis** tutorial. In this tutorial we covered how to create a timing diagram, drawing simple signals and clocks, basic timing analysis with delays and setups, and advanced drawing techniques using equations and buses.



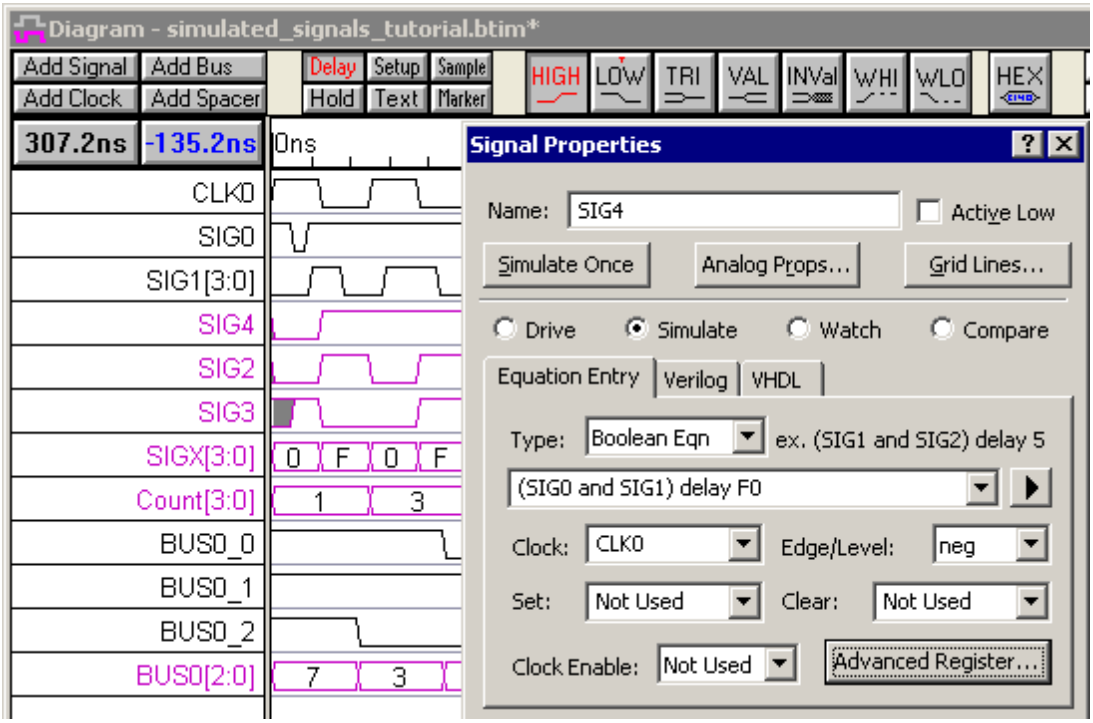
What to do next:

- If you will be doing lots of timing analysis, you need to read Section 5.1 Delays in the Timing Diagram Editor menu to find out about delay correlation and how delay times are calculated.
- If you will need to make timing diagram documentation, then do the Display and Documentation 53 tutorial.
- If you are purchasing WaveFormer Pro or Data Sheet Pro, then do the Simulated Signals 36 tutorial to discover the fastest way to generate timing diagrams.
- If you are working with analog signals, then do the Analog Signals 70 tutorial.
- If you are going to be working with lots of timing parameters, do the Parameter Libraries tutorial 90 to learn about making libraries.

Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 2: Simulated Signals

Simulated Signals reduce the amount of time needed to draw and update a timing diagram, because the waveform is described using a Boolean or registered logic equation. With Simulated Signals you will no longer have to figure the output of a combinational circuit or calculate the critical path of a synchronous circuit by hand. WaveFormer Pro has an internal interactive simulator that supports multi-bit equations with true min-max timing, unlike traditional simulators that can only represent single-valued delays. This tutorial contains some simple examples of Boolean and registered logic equations that showcase the simulator's capabilities.



To do this tutorial, you will need WaveFormer Pro or a higher level product. SynaptiCAD has also included Simulated Signals with the VeriLogger and TestBencher products, even though they have a built-in Verilog simulator, because this feature makes it easier to generate test benches and timing diagrams. In WaveFormer, it is the backbone of the timing analysis and design features.

This tutorial assumes that you are able to draw signals and can add delays, setups, and holds to those signals. We recommend that beginners start with the Basic Drawing and Timing Analysis Tutorial 10 to learn the basics of timing diagram editing before attempting this tutorial.

(TD) 2.1 Setup for Simulation

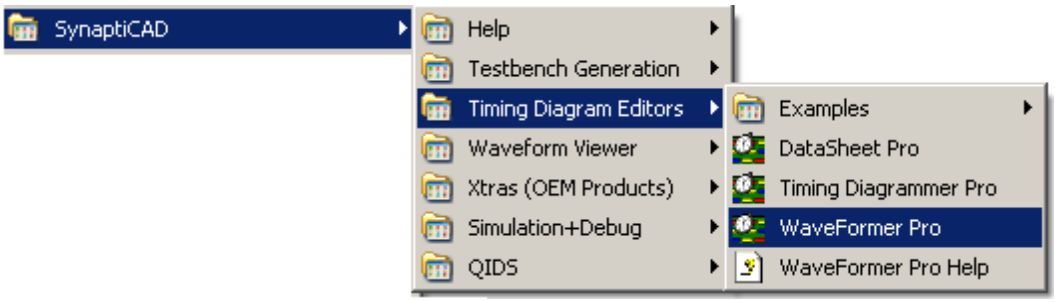
In the next few sections we will simulate signals using Boolean and registered logic equations. The inputs to a simulated signal are other drawn signals, so in this section we will create a timing diagram and a free parameter that we will use in the subsequent steps.

Run WaveFormer Pro or higher:

- Run WaveFormer Pro, DataSheet Pro, VeriLogger, or one of the more advanced products. If you are evaluating Timing Diagrammer Pro or one of our Viewers and you would like to learn about the simulation features, close the program and restart the evaluation version in

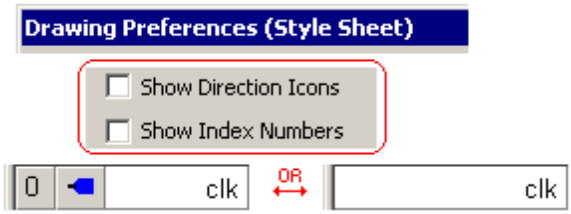
Copyright © 2014, SynaptiCAD Sales, Inc.

WaveFormer Pro mode.

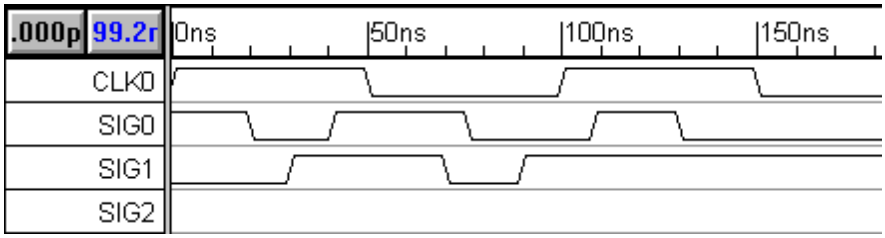


Create a Timing Diagram:

- Choose **File > New Timing Diagram** menu to open an empty timing diagram window.
- Hide the direction and index columns in the diagram window by choosing **Options > Drawing Preferences** to open the dialog. Then uncheck **Show Direction Icons** and **Show Index**.



- Sketch the following timing diagram. Clock **CLK0** has the default 100ns period. Just approximately sketch the waveforms for SIG0 and SIG1; exact edge placement is not necessary for this tutorial. Leave SIG2 blank, because it will be the simulated signal.



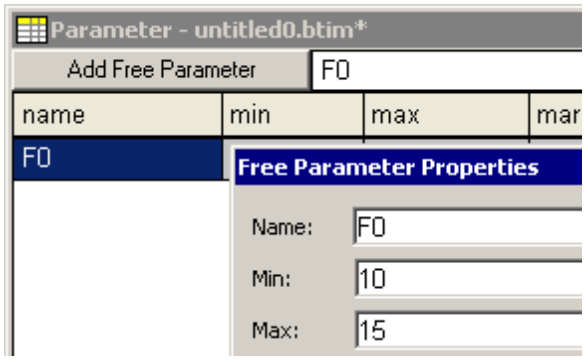
Create a Free Parameter:

We will also be experimenting with the min and max timing features of the Boolean equations, so create a Free Parameter to use in the equations.

- In the *Parameter* window, press the **Add Free Parameter** button to add a free parameter **F0**.



- Double-click on **F0** to open the *Parameter Properties* dialog.
- Enter a min time of **10**, and a max time of **15**, then press the OK button to close the dialog.



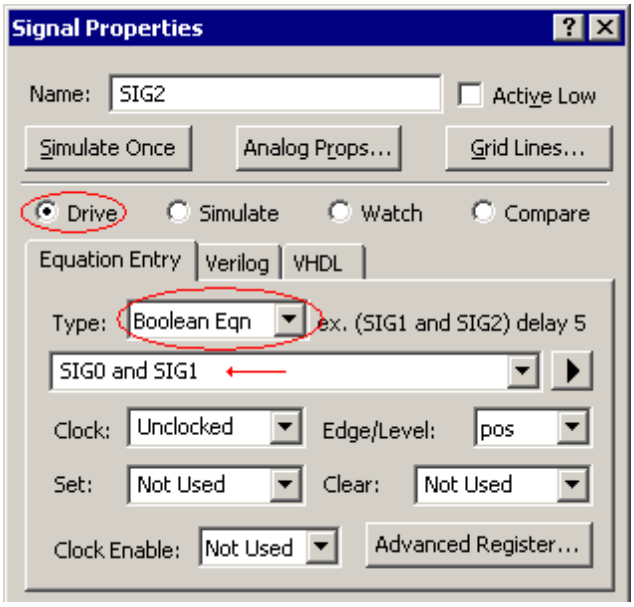
Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 2.2 Simulate a Boolean Equation

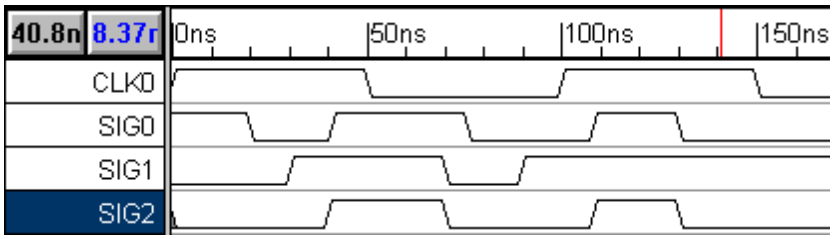
A simulated signal is created by adding a Boolean equation to the *Signal Properties* dialog for that signal. The dialog accepts Boolean equations in either VHDL, Verilog, or SynaptiCAD's enhanced equation syntax. The SynaptiCAD format supports the following operators: **and** or **&**, **or** or **|**, **nand**, **nor**, **xor** or **^**, **not** or **~** or **!**, and **delay**. We will cover the **delay** operator in the next section.

Simulate a Boolean equation once:

- Double click on the **SIG2** signal name to open the *Signal Properties* dialog. Move the dialog so that you can see the dialog and the 3 signals at the same time.
- Enter **SIG0 and SIG1** into the edit box below the **Boolean Eqn** type box (signal names are case sensitive).
- By default, all signals are **Drive** signals that will only simulate when the user presses the Simulate Once button.



- Click the **Simulate Once** button at the top of the dialog and watch the signal draw itself. Notice that SIG2 is the result of the Boolean Equation "SIG0 and SIG1". SIG2 is drawn in black to indicate that it will not re-simulate automatically.

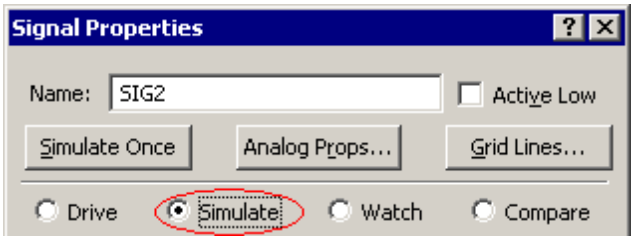


Continuously Simulate the Boolean Equation:

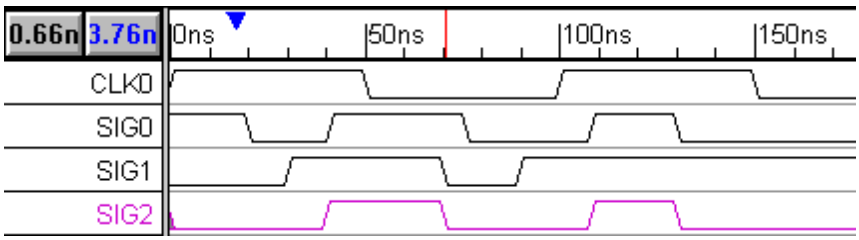
- First make sure the program is in the continuously simulate mode:
  - If you are using WaveFormer, then check the **Options > Diagram Simulation Preferences** menu to make sure that the **Continuously Simulate** box is checked. WaveFormer does not have the Auto Run/Debug Run button.
  - If you are using VeriLogger Pro or TestBencher Pro, make sure that the program is in **Auto Run** simulation mode. ***Debug Run*** mode will not continuously update signals. The Auto Run/ Debug Run simulation mode button is located on the simulation toolbar, in the upper left of the window below the Project menu.

Copyright © 2014, SynaptiCAD Sales, Inc.

- To make the signal continuously simulate, check the **Simulate** signal type button.



- Notice that the SIG2 is now drawn in purple. This color means that the signal is being continuously simulated, and changes in the input waveforms cause automatic re-simulations.



- Move some of the edges on SIG0 and SIG1 and watch SIG2 re-simulate. (Notice that you cannot drag and drop SIG2's signal edges because they are calculated edges).

(TD) 2.3 Boolean Equations with Delays

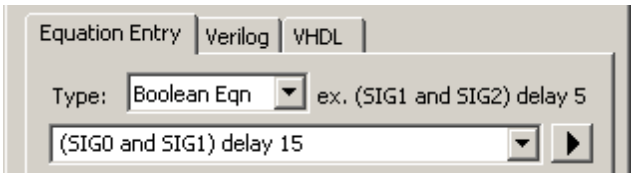
Next we will modify the Boolean equation to take into account the propagation delay through the AND gate. First we simulate a simple 15ns delay, then we will simulate a min/max delay. The SynaptiCAD **delay** operator takes a signal on the left, and a time or parameter name on the right, and returns a signal. If a parameter name is used on the right hand side of the delay operator, then the equation will simulate true min/max timing. This true min/max timing is the main advantage that SynaptiCAD's format has over the VHDL or Verilog format.

Simulate a simple delay:

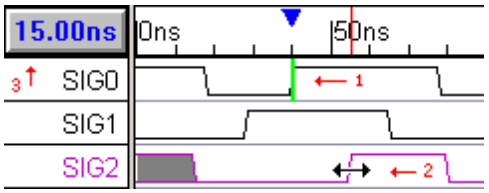
- Add a **15ns** delay to the **SIG2** Boolean equation. The box accepts Verilog, VHDL, or SynaptiCAD syntax, so these equations are all equivalent to each other.

```
#15 (SIG0 & SIG1)
(SIG0 and SIG1) after 15
(SIG0 and SIG1) delay 15
```

- Press the **Apply** button at the bottom of the dialog to notify the simulator about the change in the equation.



- Verify that **SIG2** is 15ns delayed, by first selecting an input edge then moving the mouse over the resulting edge on **SIG2**. The blue delta read out will say 15ns.



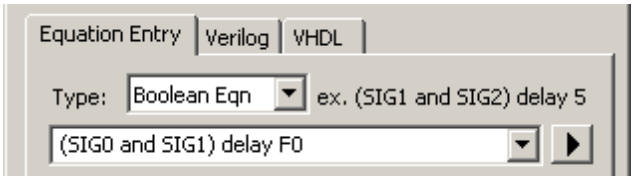
Simulate a true min/max delay using SynaptiCAD syntax:

The most powerful feature of a timing diagram editor is the ability to display min/max timing using the grey uncertainty regions. To make the Simulated Signals support min/max timing, we created the

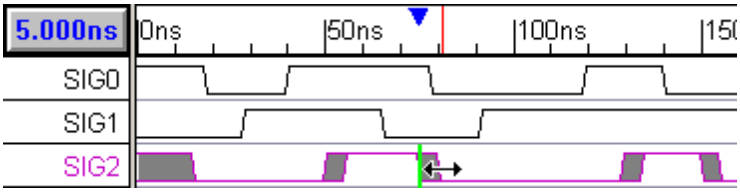
Copyright © 2014, SynaptiCAD Sales, Inc.

SynaptiCAD **delay** operator, because the delay operators in VHDL and Verilog only support a single delay.

- Edit the equation so that the delay references the free parameter **F0** then press the **Apply** button to notify the simulator about the change in the equation.



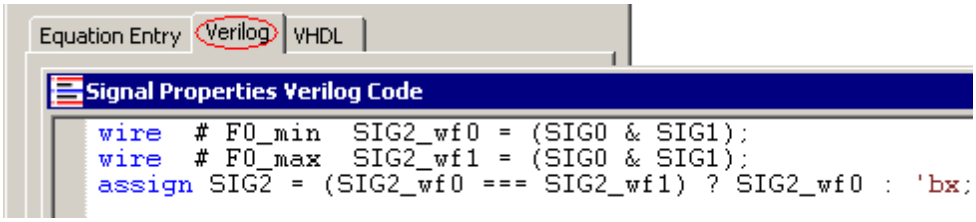
- Notice that SIG2 has grey uncertainty regions that are 5ns wide (F0.max - F0.min).



View the HDL code that models the Boolean equation:

The timing diagram editor takes the Boolean equation and generates Verilog or VHDL code necessary to perform the equation. You can edit this code directly to perform more complex functions. The tool ships with an embedded Verilog simulator that executes the code, so if you change VHDL code you will have to provide the tool with a VHDL simulator. The manual explains how to configure for a different simulator.

- Press the **Verilog** tab to open an editor window that displays the generated Verilog code. Do not change the code now.

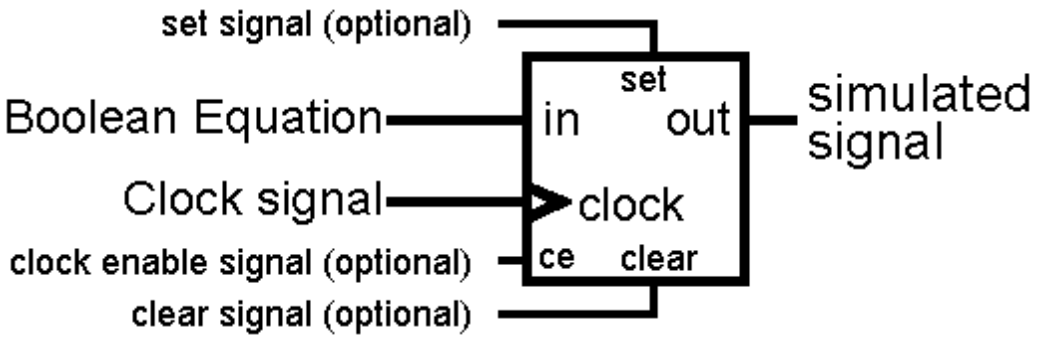


- This example demonstrated true min/max simulation, however Min-Only and Max-Only simulations can be performed by changing the selection in the **Timing Model** drop-down list of the *Simulation Preferences* dialog box. The *Simulation Preferences* dialog can be opened using the **Options > Diagram Simulation Preferences** menu option. The **Timing Model** drop-down list is in the upper right corner.

(TD) 2.4 Register and Latch Signals

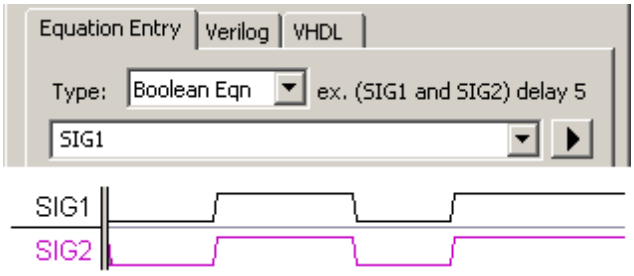
The Interactive Simulator can register or latch the result of a Boolean equation. This circuit is similar to most FPGA cells and can model a large number of components. Below is a figure of the register that a Simulated Signal models. If no clocking signal is chosen, then the Boolean equation goes straight to the signal output as shown in the previous sections. Note: setting the MSB/LSB fields in the Signal Properties dialog will "parallelize" the circuit, allowing multi-bit registers such as counters, shifters, etc. to be modeled.

Copyright © 2014, SynaptiCAD Sales, Inc.

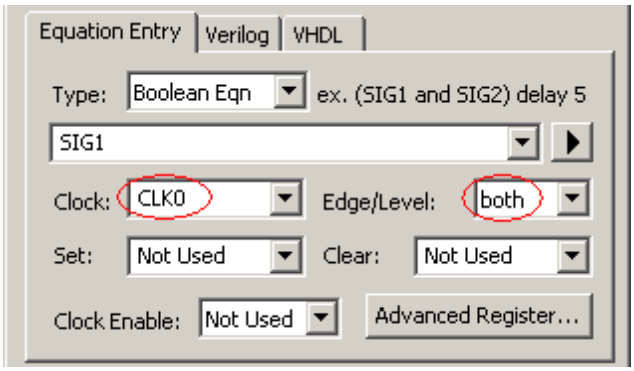


Experiment with the register and latch equations:

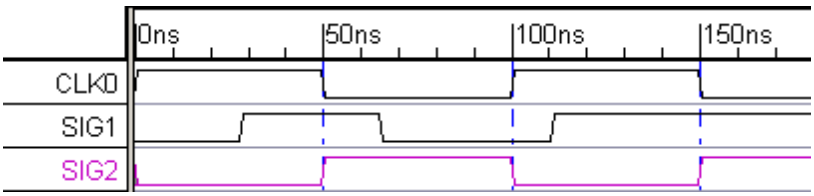
- Change the **SIG2** equation to just one term ***SIG1*** and press the **Simulate Once** button. SIG2 should be an exact copy of SIG1.When we register SIG2 you can visually compare it to SIG1 to see the effects of the register.



- Set the clock control to **CLK0**. Choosing a clock brings in the register/latch model to buffer the Boolean equation.
- Set the edge/level control to **both**, to indicate that both the rising and falling edges of the clock are triggering edges. Since it is edge triggered a register circuit will be created rather than a level sensitive latch.

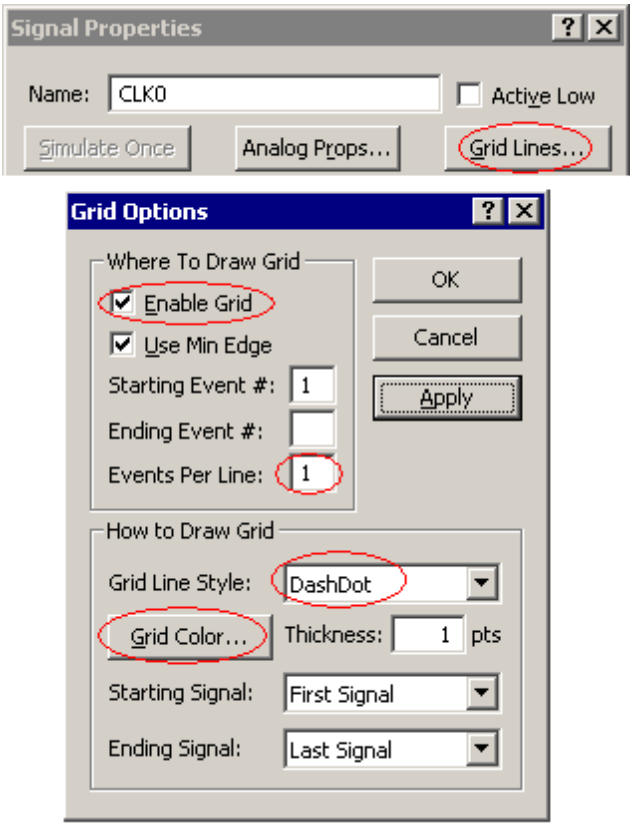


- Click the **Simulate Once** button to simulate the circuit. Notice that SIG2 only transitions when CLK0 has a positive or negative edge transition (move some edges on SIG1 to verify this).



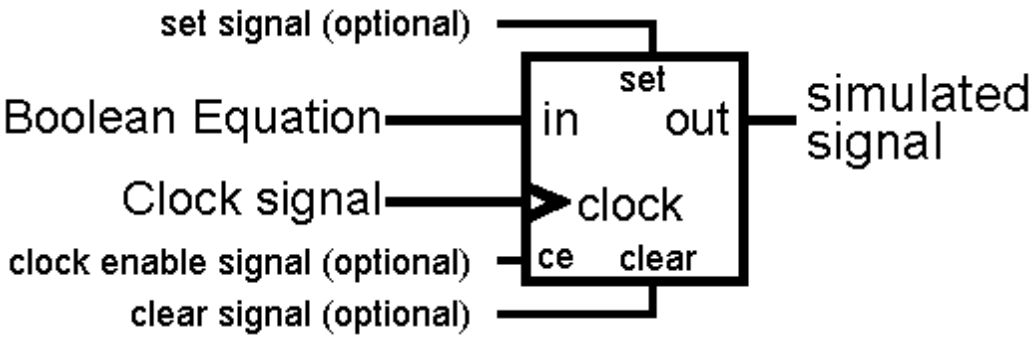
Copyright © 2014, SynaptiCAD Sales, Inc.

- To make the diagram look like the above picture, we hid **SIG0** because it is not being used.
- We also added **Grid Lines** to every edge of the clock. To do that double click on the clock name to open the *Signal Properties* dialog and press the **Grid Lines** button. This opens the *Grid Options* dialog. Play with the controls and hit **Apply** until you get an image that you like.



(TD) 2.5 Set and Clear Lines

The **Set** and **Clear** lines are useful when defining circuits whose initial value needs to be specified. In this example we demonstrate how to design a **divide by 2 circuit** using a negative edge triggered register with an asynchronous active-low set line.



Copyright © 2014, SynaptiCAD Sales, Inc.

Use the Set line to define an initial state:

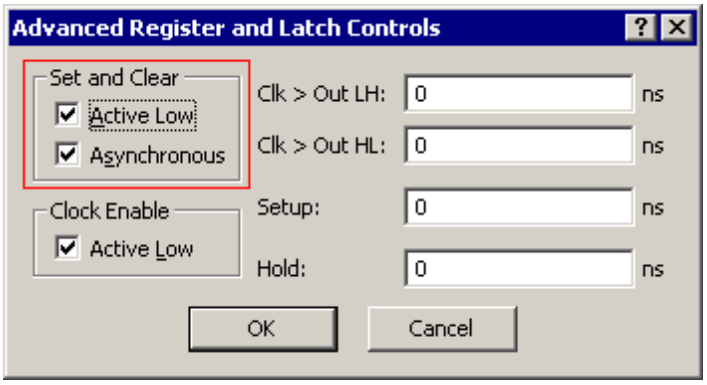
- Click the **Add Signal** button to create a new signal named **SIG3**. Then double click on the signal name to open the *Signal Properties* dialog.
- Check the **Simulate** button.
- Type **!SIG3** into the *Boolean Equation* edit box (it references itself).
- Choose **CLK0** from the *Clock* drop down list box.
- Choose **neg** from the *Edge/Level* box.



- Notice that the waveform for **SIG3** is completely gray but that the status bar (in the lower right corner of the window) reports **Simulation Good**. This is because **SIG3**'s Boolean equation references itself but it does not provide the simulator with a known start state.



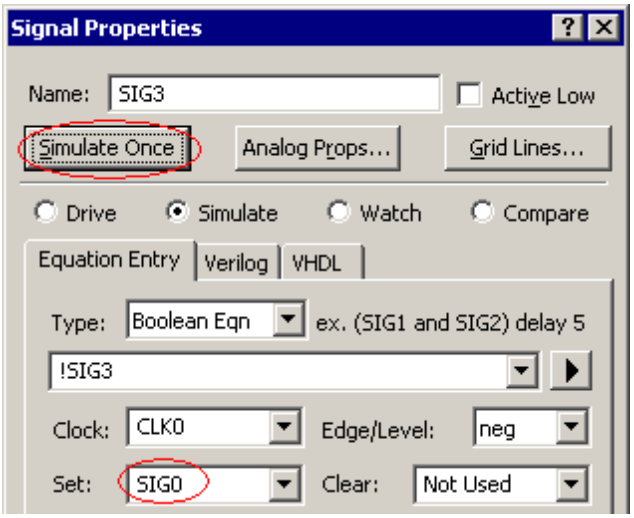
- Press the **Advanced Register** button to open the *Advanced Register and Latch Controls* dialog. Notice that the register and latch propagation, setup, and hold times, clock enable, and set/clear options are set here. Tip: the Global defaults are set using the **Options > Simulation Preferences** menu.



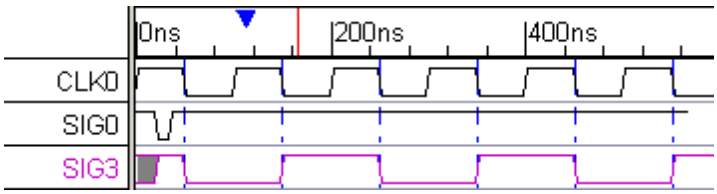
- Make sure the **Active Low** and the **Asynchronous** check boxes in the *Set and Clear* section are checked. Click **OK** to close the dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Choose **SIG0** from the *Set* drop down list box.
- If you hid SIG0 in a previous section, choose the **View > Show and Hide Signals** menu to show SIG0.
- Press the **Simulate Once** button to notify the simulator of the change in the model.



- Notice that **SIG3** now has a simulated waveform. Redraw **SIG0** so that it goes **low** early in the timing diagram, and then stays **high** for four or five clock cycles.



- Experiment with **SIG0** to see how the active low set line affects the operation of the flip-flop. Remember that we set the model to have an asynchronous low set signal.
- To make the above diagram, we hid the unused signals. We also changed the clock grid so that the **starting event** was 2 and there were 2 **events per line**, that way we got grid lines on just the negative edges.

(TD) 2.6 Multi-bit Equations

The Interactive Simulator can automatically generate multi-bit equations for the register, latch and combinatorial logic circuits. To convert a register or latch circuit into a multi-bit signal, change the MSB of the input signal and the MSB of the register or latch. If the sizes of the signals do not match, WaveFormer maps as many LSB's as it can. The following example uses only a simple equation to demonstrate the LSB mapping feature, however multi-term Boolean equations are completely supported.

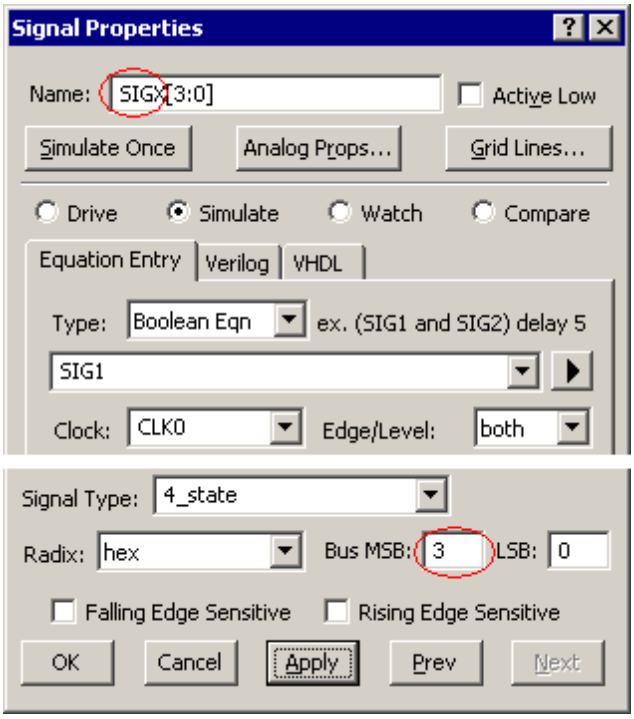
Create a Multi-bit Signal by changing the MSB setting:

- Create a copy of **SIG2**. Click on the **SIG2** name in the Label window to select it. Select the **Edit > Copy Text and Signals** menu option to copy the signal, then the **Edit > Paste** option to paste the signal. There are now two signals named **SIG2** in your diagram.

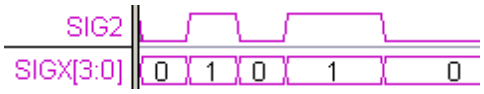


Copyright © 2014, SynaptiCAD Sales, Inc.

- Double click on the bottom **SIG2** to open the *Signal Properties* dialog, and rename the signal to **SIGX**.
- Type **3** in the **Bus MSB** edit box. This will make **SIGX** a 4-bit signal and add a **[3:0]** to the end of the name.
- Press the **Apply** button to notify the simulator of the changes.

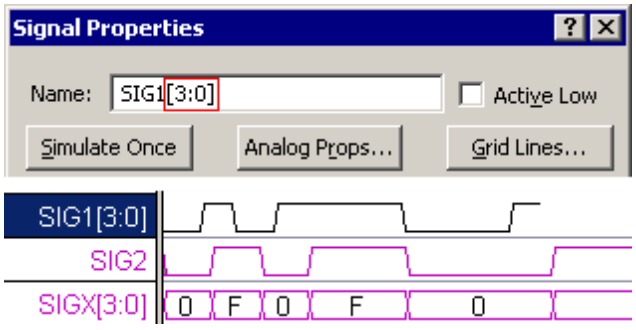


- **SIGX**'s waveform is now drawn as a bus with a 4 bit binary display. Only the LSB of **SIGX** is working because the input signal **SIG1** is a single bit. Compare **SIG2** and **SIGX** and verify that they are the same values.



Change the input signal to a multi-bit signal:

- Double-click on **SIG1** to open the *Signal Properties* dialog, and add **[3:0]** to the end of the name. This has the same effect as changing the values in the **MSB** and **LSB** edit boxes.
- Press the **Apply** button. Now all four bits of **SIGX** should be toggling between 0 and F. If the radix is in Binary, the signal will toggle1111 and 0000. The radix box is located in the lower left part of the dialog.



- If you want to further experiment with multi-bit signals, change SIG1's graphical segments to Valid regions instead of Highs and Lows. Then double click on a valid region to open the Edit Bus State dialog box. Type different hex values from 0 through F, like 5 or A, into the Virtual edit box and watch how it affects the output of SIGX and SIG2. Since SIG2 is a single bit signal it uses only the LSB of the input signals.

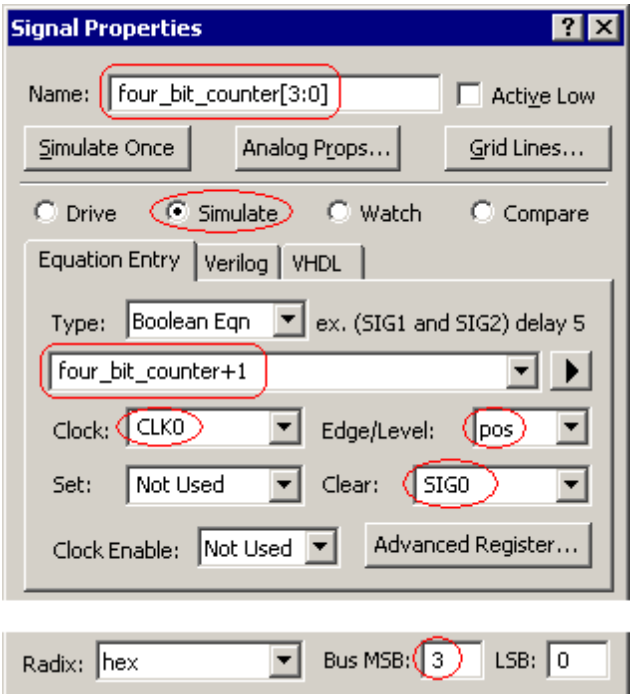
Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 2.7 Design a Multi-Bit Counter

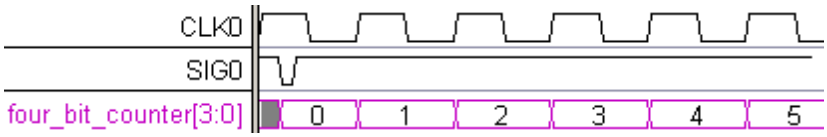
When a muti-bit signal is clocked the register/latch circuit shown will be instantiated one time for each bit of the signal. This allows you to use one signal to represent the operation of a multi-bit counter or buffer.

Use a multi-bit signal to make counter:

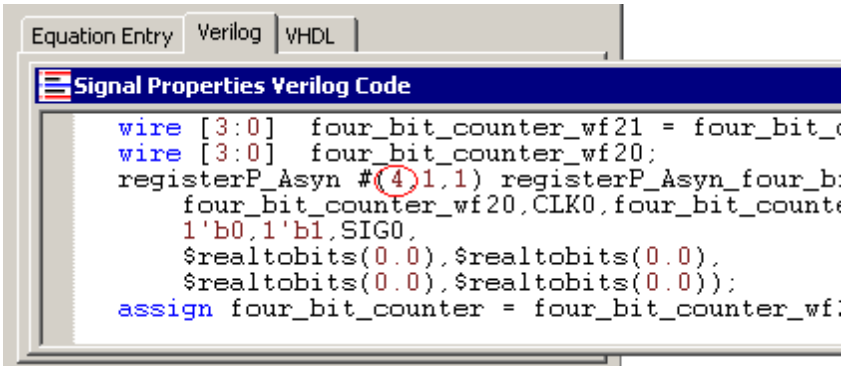
- Press the **Add Signal** button to add a new signal.
- Rename the signal to **four_bit_counter.**
- Set the type to **simulate**.
- Type in the equation **four_bit_counter +1**.
- Set the clock to **CLK0** with an trigger edge of **pos**. This will make the signal change only on the positive edges of the clock.
- Set the clear line to **SIG0**. If you press the **Advanced Register** button you can verify that the clear line with be active low and asynchronous, so that a pulse low on this line will clear the signal registers to zero.
- Set the Msb to **3** to instantiate the multiple registers.



- Press the **Apply** button to simulate the signal. Notice that the signal will be a grey unknown region until the SIG0 goes low to clear the register.



- If you press the **Verilog** tab, in the *Signal Properties* dialog you can see that the register is going to be 4 bits wide (it is the first parameter that is passed into the register).



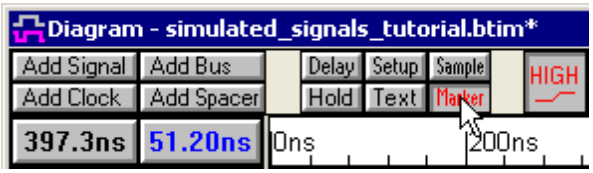
Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 2.8 End Diagram Marker Stops Simulation

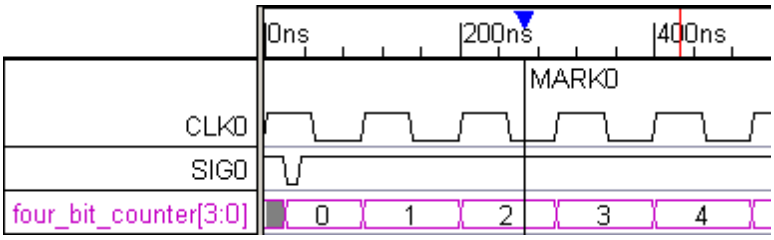
Normally the simulation will continue to the end of the last drawn signal or about one clock cycle past the drawn signal. However, the exact end of simulation can be controlled using Marker line.

Use an End Diagram Marker to control the end of simulation time:

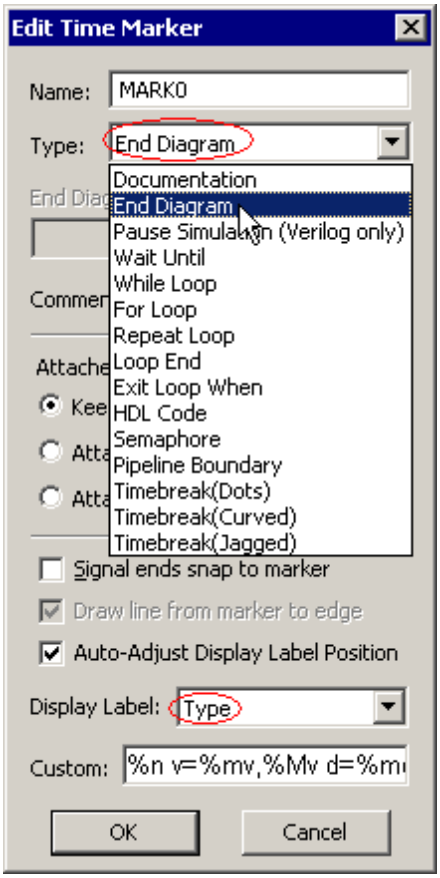
- Press the **Marker** button so that right clicks will add marker lines to the diagram.



- Right click inside the diagram window to add a **Marker** line. By default all new markers are just documentation lines that will not effect the timing or simulation of the diagram.



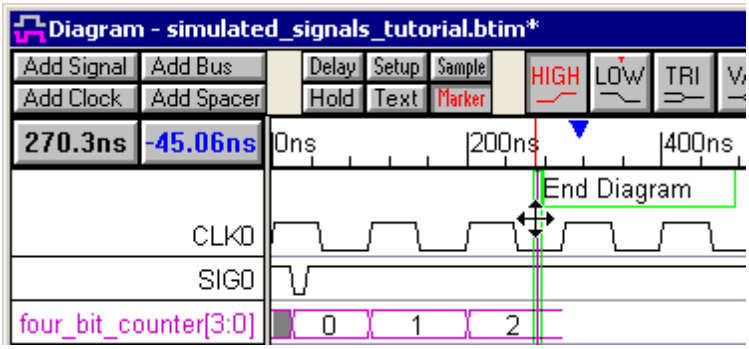
- Double click on **MARK0** to open the *Edit Time Marker* dialog.
- Set the type to **End Diagram**. But also notice all the other types available. The *timebreak* types compress time and hide parts of the dialog. The loop markers are used by TestBencher and Reactive Test Bench generation to create complex test bench code.
- Set the display label to **Type**. This will make the marker display its type instead of its name. Also notice all the display label options to control exactly what the marker displays.
- Press the **Ok** button to close the dialog



- Notice that the simulation ends at the clock cycle after the End Diagram Marker. Grab the

Copyright © 2014, SynaptiCAD Sales, Inc.

marker with the mouse and drag it around to control the simulation end.



(TD) 2.9 Behavioral HDL Code

In addition to the simulation of Boolean and registered logic circuits, SynaptiCAD products can simulate behavioral Verilog code, and if you provide a VHDL simulator it can also simulate behavioral VHDL code. The behavioral code is entered directly into the **Verilog** or **VHDL** tabs in the *Signal Properties* dialog, instead of using the **Equation Entry** tab that we used for the Boolean equations in the last sections. There is also a template feature that generates code from a Boolean equation and allows you to modify the generated code.

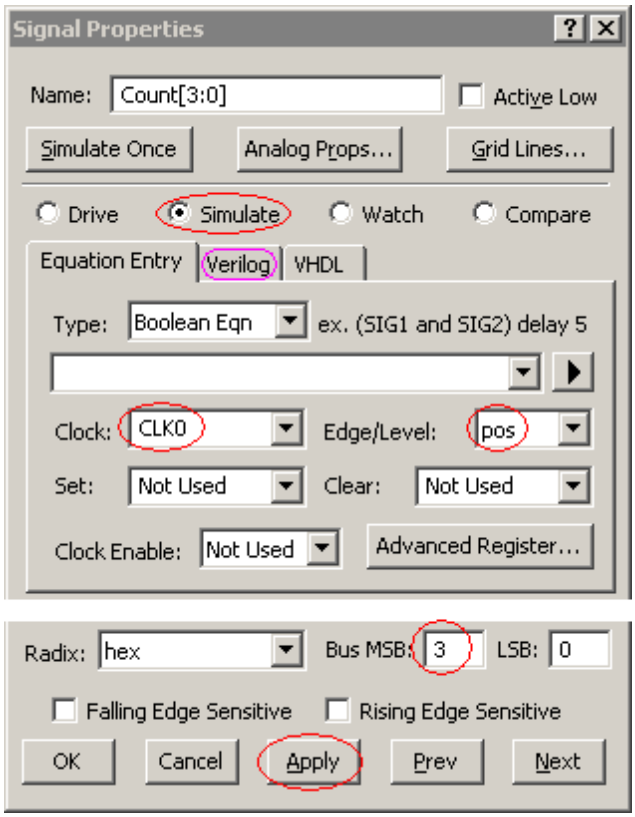
In this section we will use a register template as a starting point to build a circuit that asynchronously counts the number of edges that occur on SIG1 and synchronously presents the total number of edges on the positive edge of the clock.

Copyright © 2014, SynaptiCAD Sales, Inc.

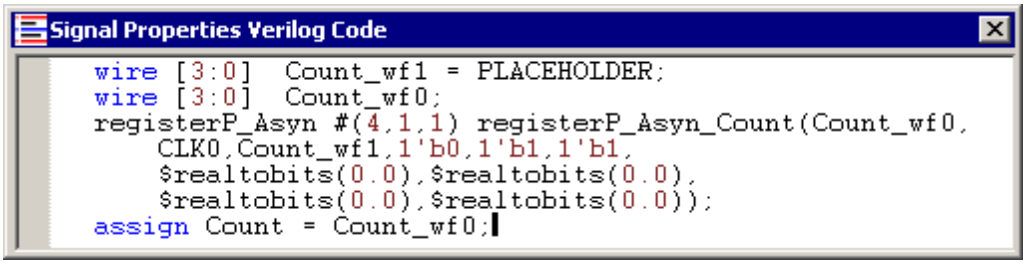
Use a partially defined signal to generate PLACEHOLDER code:

- Add a new signal named **Count**.
- Select the **Simulate** type setting.
- Set the clock to **CLK0** with an edge/level of **pos**.
- Set the MSB to **3**.
- Press the **Apply** button to apply the changes and generate the code. Since there is no Boolean equation for the signal, this is also going to generate a compile error and all the simulated signals in the diagram will go grey. If no compile error happens, then choose **Options > Diagram Simulation Preferences** menu and check the **Continuously Simulate** box.

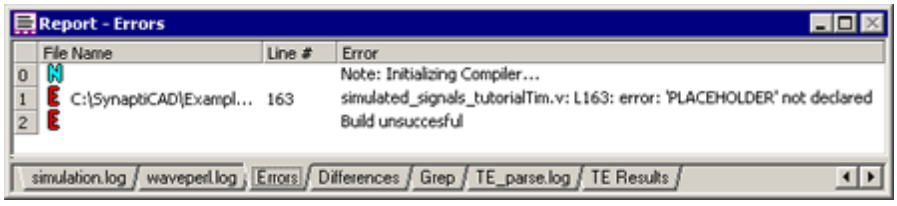




- Next, press the **Verilog** tab to open an editor window containing the the template code. The internal wire names Count_wf*** will vary depending on how many signals you have simulated.



- The **registerP_Asyn** line instantiates (defines an instance of) a 4 bit positive-edge-triggered register of the type used by the logic wizard. This register takes PLACEHOLDER as an input and outputs a synchronized version on **Count**.
- The auto generated variable **PLACEHOLDER** is undefined and will not simulate. If a Boolean equation was defined for the circuit, it would replace the PLACEHOLDER variable. This error will be displayed in the Report window under the error tab.



Copyright © 2014, SynaptiCAD Sales, Inc.

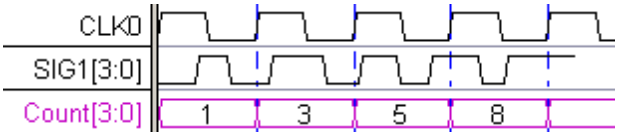
Add behavioral code to the generated code:

We will use the PLACEHOLDER variable to store the edge count. First we will define PLACEHOLDER, give it an initial starting value, then define an ***always*** process that triggers each time the SIG1 changes. Since Count is buffered by a positive edge triggered register, it will not display the PLACEHOLDER value until the positive edge of the clock.

- Either copy-and-paste or type the first four lines the below code directly into the *Signal Properties Verilog Code* window (add the bold lines):

```
reg [3:0] PLACEHOLDER;
initial PLACEHOLDER = 0;
always @(SIG1)
  PLACEHOLDER = PLACEHOLDER + 1;
wire [3:0]  Count_wf1 = PLACEHOLDER;
wire [3:0]  Count_wf0;
registerP_Asyn #(4,1,1) registerP_Asyn_Count(Count_wf0,
      CLK0,Count_wf1,1'b0,1'b1,1'b1,
      $realtobits(0.0),$realtobits(0.0),
      $realtobits(0.0),$realtobits(0.0));
assign Count = Count_wf0;
```

- Click the **Apply** radio button. Verify that Count is counting the edges of SIG1. The new edge count is presented on each positive edge of CLK0. The Count starts at one because there is a 1'bz to 1'b0 transition at time zero on SIG1.



Code explanation:

The code that you just entered is behavioral Verilog code.

- The first line defines PLACEHOLDER as a 4-bit register. PLACEHOLDER needs to be defined as a register rather than a wire in this case because it must "remember" its value. Verilog wires don't remember their values so they must be constantly driven to retain their value.
- The second line initializes the value of PLACEHOLDER to 0 when the simulator first runs.
- The third and fourth lines contain an always block (note for VHDL users: these work like VHDL process blocks). Whenever **SIG1** changes state, the always block will execute, incrementing PLACEHOLDER.
- The rest of the lines consist of the automatically generated template code that instantiates the synchronizing register.

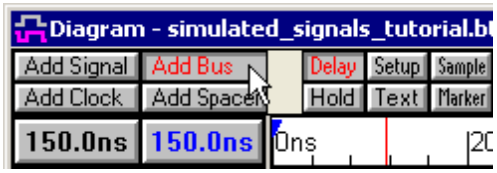
(TD) 2.10 Simulated Bus Signals

Simulated Buses are similar to Group Buses. The primary difference is that the bus is generated using a Boolean Equation. A simulated bus can be referenced in another signal's Boolean equation, (group buses cannot). Also, TestBencher will generate a Boolean equation for the timing transaction so that the simulated bus can include input signals as member signals.

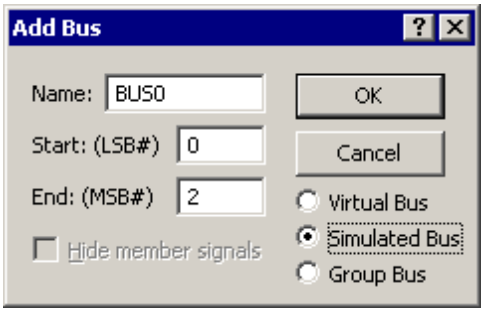
Copyright © 2014, SynaptiCAD Sales, Inc.

Create a Simulated Bus:

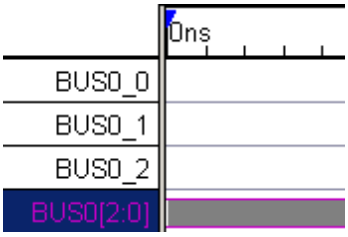
- Make sure that no signals are selected, and press the **Add Bus** button to open the *Add Bus* dialog. Note, if a signal was selected a different dialog will open and you cannot add a simulated bus from there.



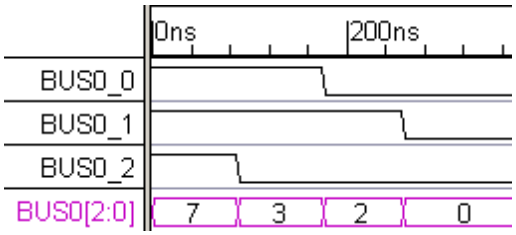
- Set the **LSB** to **0** and the **MSB** to **2**.
- Select the **Simulated Bus** radio button.
- Press the **OK** button to close the dialog and create the bus and member signals.



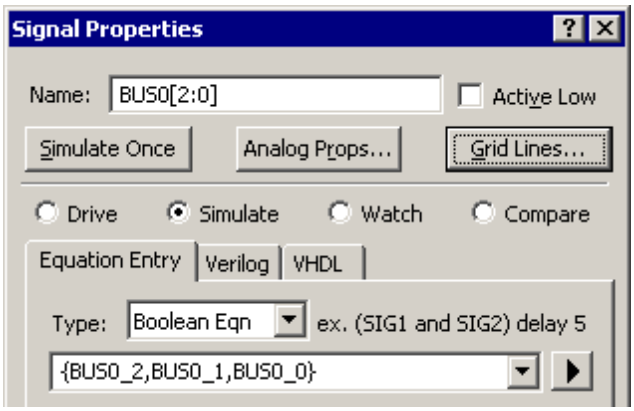
- BUS0"s waveform will be grey because none of the member signals are defined.



- Draw on the member signals and see the bus simulate.



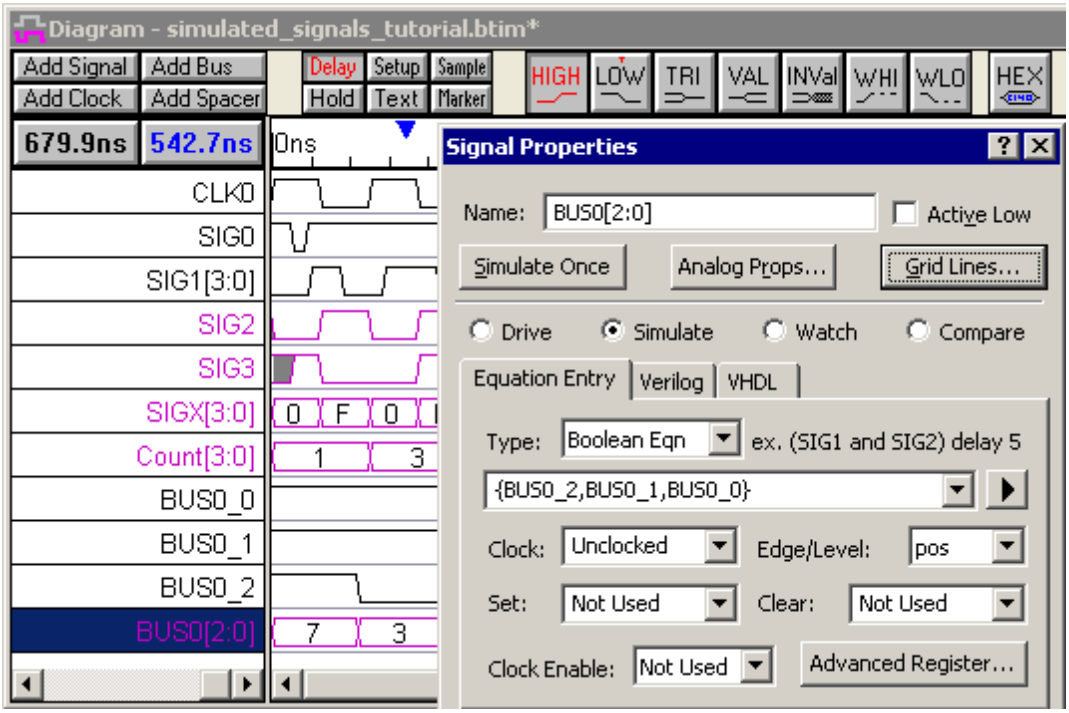
- Double click on **BUS0[2:0]** label to open the *Signal Properties* dialog. Notice that the bus is defined as a concatenation of the member signals.



Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 2.11 Summary of Simulated Signals Tutorial

Congratulations! You have completed the Simulated Signals tutorial. In this tutorial, you have used a Boolean Equation to define the waveform of a signal, and experimented with the **delay** operator and multi-bit equations. You have also worked with the internal register and latch circuits and the template function for generating a starting point for writing behavioral code. And fininally you have generated a simulated bus.



For more information on simulated signals and the internal simulator look at:

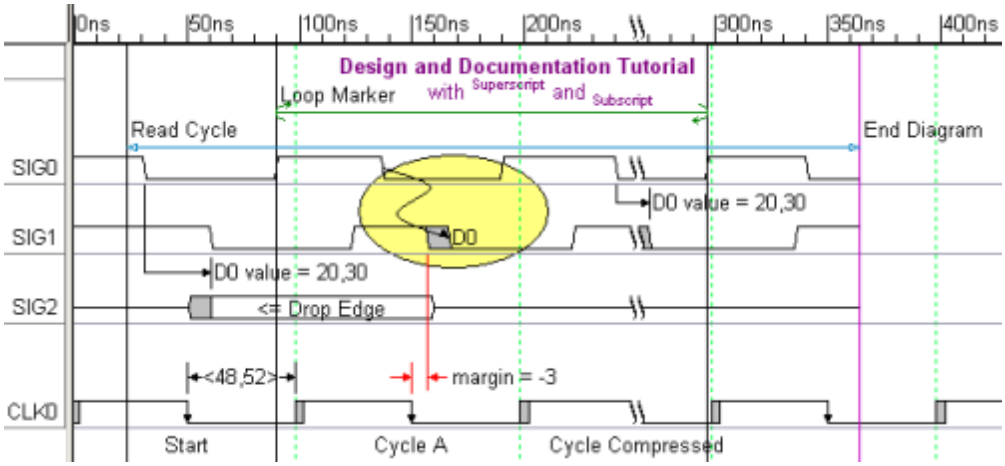
- Chapter 4: Simulated Signals and VHDL and Verilog Export in the Timing Diagram Editor manual.
- Advanced Modeling and Interactive Simulation 98 Tutorial demonstrates how to model a complex circuit using external models, behavioral HDL code, and incremental simulation techniques.

If you currently own Timing Diagrammer Pro or one of the SynaptiCAD Viewers, then these products can be upgraded to WaveFormer Pro or Data Sheet Pro so that you will have access to the Simulated Signal features.

Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 3: Display and Documentation

This tutorial introduces techniques for controlling the display of parameters, clocks, waveforms, markers and text objects. These techniques will allow you to control exactly what your timing diagrams look like and what information is displayed.



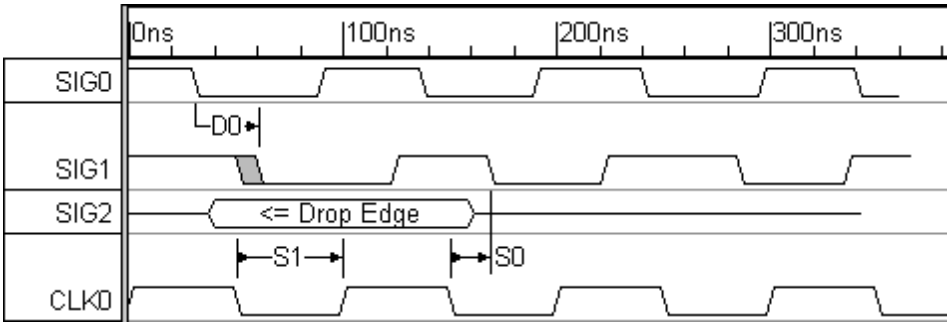
It is recommended that you become comfortable drawing waveforms and adding parameters before you begin this tutorial by first performing the Basic Drawing and Timing Analysis 10 tutorial.

(TD) 3.1 Setup for the Tutorial

Since this tutorial focuses on how to make the timing diagram look different, we will save some time by loading a pre-drawn timing diagram.

Load the starting timing diagram for this tutorial:

- Open the file **tutdocstart.btim** in the **SynaptiCAD\Examples\TutorialFiles \DisplayAndDocumentation** directory.

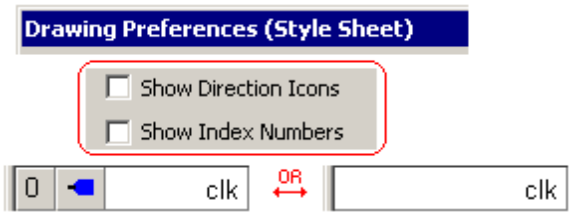


- Select the **File > Save As** menu option, and save this file as **mystart.btim**.

Copyright © 2014, SynaptiCAD Sales, Inc.

Hide the Direction and Index Columns in the Label window:

- Choose the **Options > Drawing Preferences** menu to open the dialog. Then uncheck **Show Direction Icons** and **Show Index**.



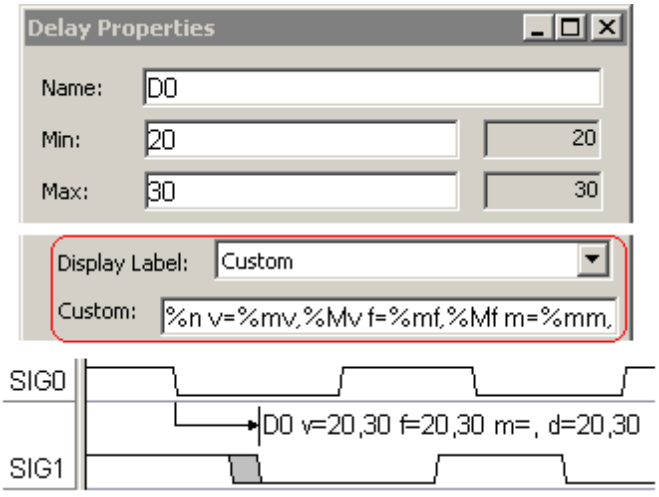
(TD) 3.2 Parameter Display Strings

A Delay, Setup, Hold, or Sample parameter can display any of its attributes like name, times, distances, margin, comment string or any combination of attributes. A parameter's display is controlled by the the **Display Label** and **Custom String** boxes in the *Parameter Properties* dialog. Even though all parameters have these features, usually Setups and Holds are used for diagram annotation and distance measurements, because these parameters monitor state information instead of forcing edges like delay parameters. In this section, we will edit the three parameters in the diagram and write some custom strings.

Display anything using a Parameter Custom Strings

A parameter label can be made to show more than one attribute or to show a custom string of characters and attributes using the **Custom** string in the *Parameter Properties* dialog.

- Double click on **D0** delay parameter to open the *Parameter Properties* dialog.
- Select **Custom** from the **Display Label** drop-down list box. This will cause the string in the *Custom* edit box to be displayed as the parameter's label. The default custom string is a little messy to look at, however it contains all of the control codes so you don't have to remember them. When you want to make a custom label just edit the default string.



- Compare the default **Custom** string to the label that is displayed in the diagram. The default custom string should be:

```
%n v= %mv,%Mv f=%mf,%Mf m=%mm,%Mm d=%md,%Md %c
```

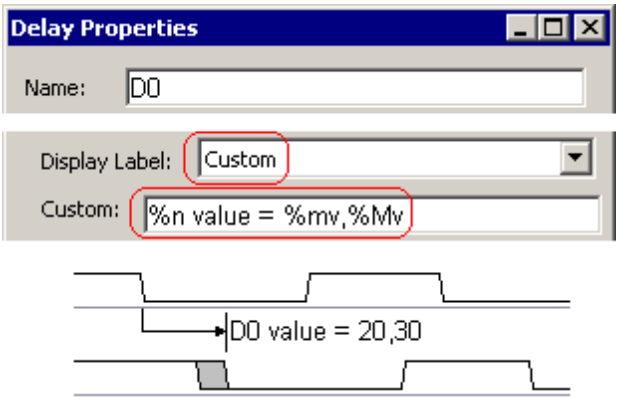
- Notice the codes that start with a **%** character followed by one or two letters are replaced in the label and all other text is just shown. The control codes are: name (**%n**), value (**%mv, %Mv**), formula (**%mf, %Mf**), margin (**%mm, %Mm**), distance (**%md, %Md**), and comment (**%c**). The lower case **m** means minimum, and the upper case **M** means maximum. The example parameter D0 is a delay, so the margin control codes %mm and %Mm are blank. Also since there is no "comment" for D0 it is also blank.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Next, make the parameter label display only the parameter name and min and max values. Edit the contents of the custom string so that the string reads:

  %n value = %mv,%Mv

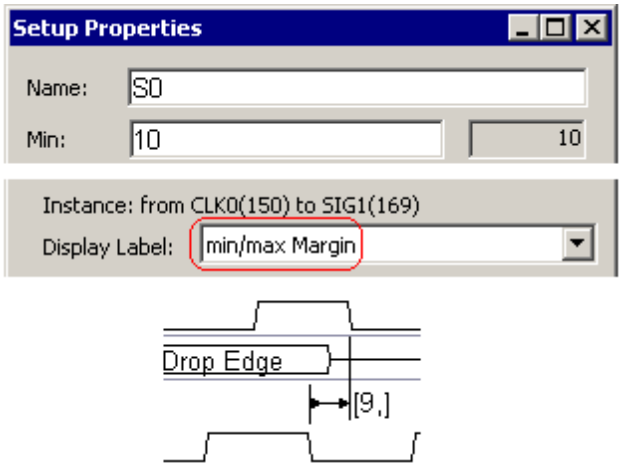
- Press the **Apply** button and D0's label will show. (LEAVE THE DIALOG OPEN).



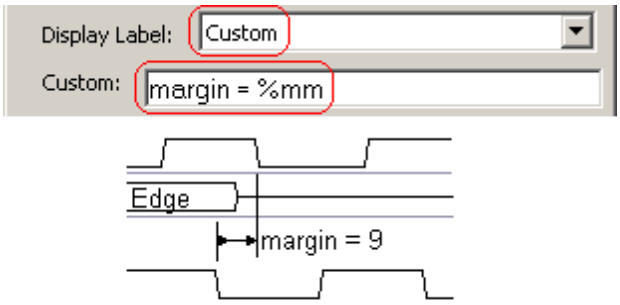
Display Margin times using a Setup or Hold:

There are also several default display labels that you can quickly use to display an attribute.

- Double-click on the setup label **S0** to open the *Parameter Properties* dialog. Arrange the dialog so that you can see the S0 in the diagram window and the dialog at the same time.
- Use the **Display Label** drop-down list box to select the **min/max Margin** display. Notice that the label for the parameter now displays **[9,]**, the min/max margin, instead of the name S0. This display will change if the setup's edges are moved. The max is blank because there is no maximum setup time specified in the parameter.



- If you do not want to display the maximum then you can create your own custom string. Here the code **%mm** will be replaced by the minimum margin time and all other text will be output as typed. (LEAVE THE DIALOG OPEN).

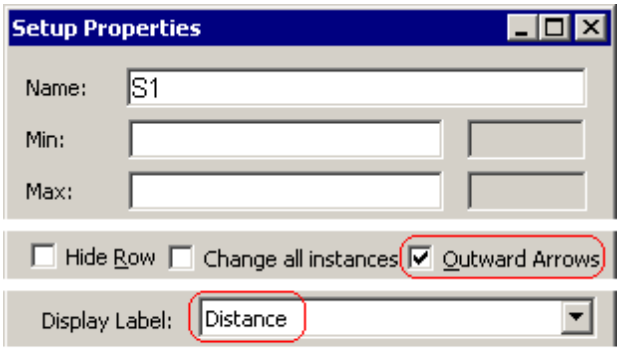


Display Distance measurements using a Setup or Hold:

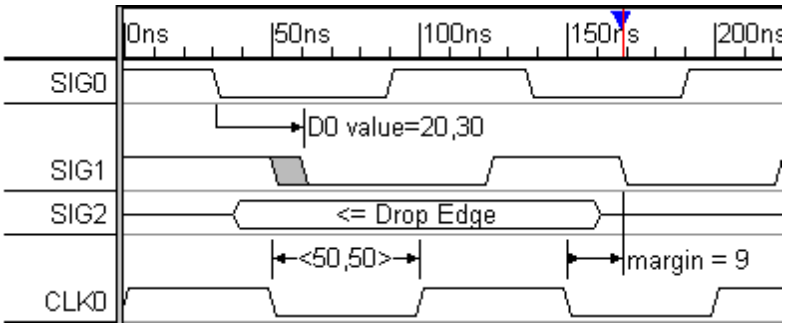
- Click the **Next** button to display the setup **S1** in the *Parameter Properties* dialog. We will just use S1 to display the distance between two edges, so we have not bothered to define min and max values for it.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Check **Outward Arrows** so that the parameter's arrows display the type of arrows that are usually used for distance measurements.
- Select the **Distance** from the **Display Label** drop-down list box. The label now shows the minimum and maximum distances between the transitions.
- This can also be achieved using a **Custom** string **<%dm, %dM>** as shown in the previous example.
- Press **Ok** to close the dialog.







Miscellaneous Parameter Information:

The default display for all parameters can be set using the **Options > Drawing Preferences** dialog box. Also individual instances or all instances of a parameter are configured depending on where the *Parameter Properties* dialog is opened. To edit an individual instance, double click on a parameter in the *Diagram* window. To configure all instances of a parameter, double click on a parameter row in the *Parameter* window.

(TD) 3.3 Repeat Parameters Across the Diagram

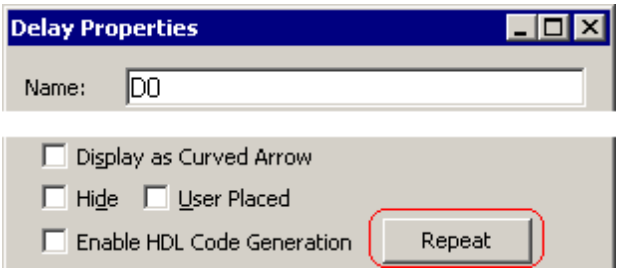
Once you have drawn a delay, setup, or hold parameter, that parameter can be automatically drawn between similar edges across the timing diagram. When the Repeat button, in the *Parameter Properties* dialog, is pushed the program will search for the next beginning edge, and add a parameter between that edge and the next ending edge. This will continue until the end of the diagram. Some caution should be taken when repeating delays because the delays cause edges to move.

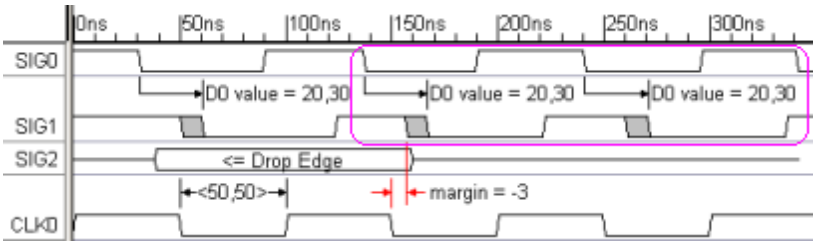
Repeat D0 across the timing diagram:

- For this demonstration arrange *Diagram* window so that you can see the entire diagram. You may need to use the zoom-in buttons.

Copyright © 2014, SynaptiCAD Sales, Inc.

- In the diagram window, double-click on **D0** to open the *Parameter Properties* dialog.
- Press the **Repeat** button. This will cause delays to be added to each of the falling edges of SIG0 that have a matching edge on SIG1.



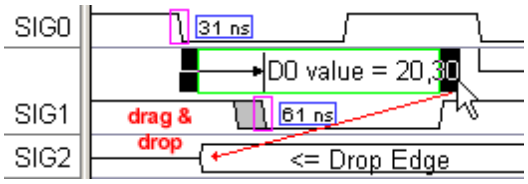


- Also notice that the margin for setup S0 is now violated and is displayed in red. This happened because the second D0 moved the edge on SIG1 that S0 is attached to.
- Close the *Parameter Properties* dialog.

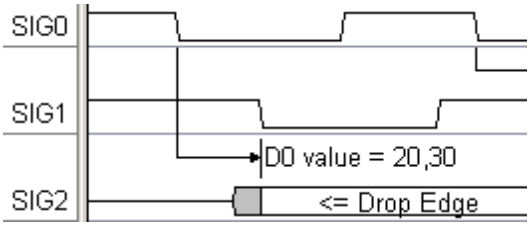
(TD) 3.4 Move Parameters to Different Signals

A delay, setup, hold, or sample parameter can be moved to a different signal transition by dragging and dropping one of the parameter end-points.

- In the Diagram window, select the first delay parameter D0 clicking on it. A selected parameter is surrounded by a rectangle with a solid handle box on either end.



- Drag and Drop the black handle the ***right side*** of the selection rectangle to the first edge on SIG2 as indicated. If the entire parameter is changing its vertical position, then you clicked on the middle of the parameter instead of a handle box.

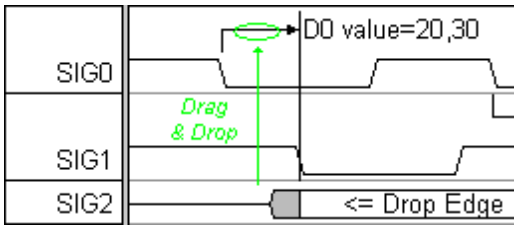


(TD) 3.5 Adjust Parameter Vertical Placement

Normally, the vertical placement for parameters on the sreen is set automatically. However, you can also place parameters at a specific height by dragging the parameter to a new position.

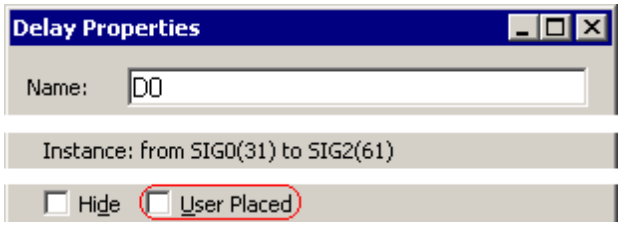
Copyright © 2014, SynaptiCAD Sales, Inc.

- Click down on the center of the first delay parameter, **D0**, and **drag** it up to a new vertical position closer to the top of the screen.
- Release the mouse button to place the parameter.

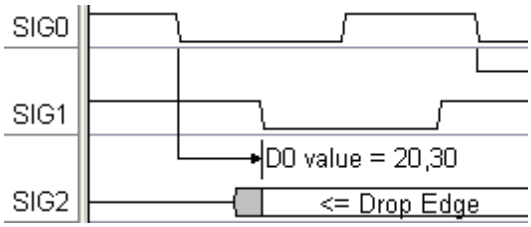


After you move a parameter, it is considered user placed and it will not be moved from that position unless you choose to move it. Any new parameters will arrange themselves around user placed parameters. To return vertical placement control to the program:

- Open **D0**'s *Parameter Properties* dialog box by double-clicking on the parameter.
- Uncheck the **User Placed** box, to return placement control to the program.



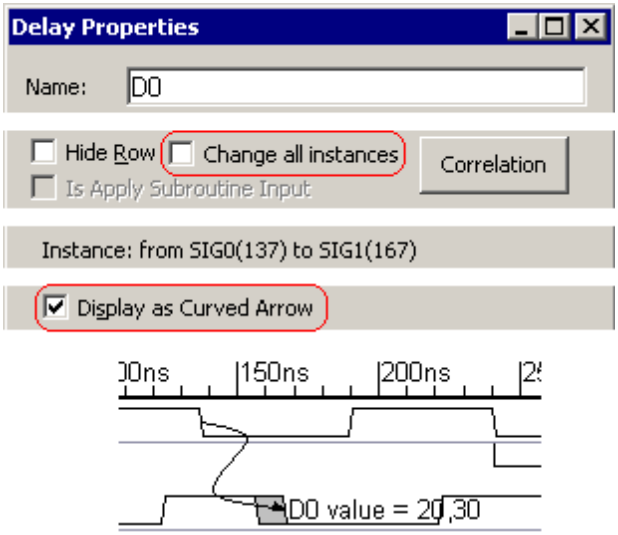
- Press the **OK** button to close the dialog and watch how the delay returns to its default position.



(TD) 3.6 Curved Parameters

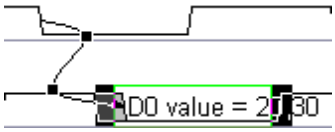
By, default parameters are drawn using straight lines, however you can make them have a curved line.

- Double click on the second D0 in the diagram to open its *Delay Properties* dialog.
- Since the **Change all instances** is unchecked all changes made in this dialog will only effect this instance of D0.
- Check **Display as Curved Arrow** to make the delay redraw as as curved arrow.

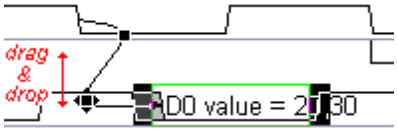


Copyright © 2014, SynaptiCAD Sales, Inc.

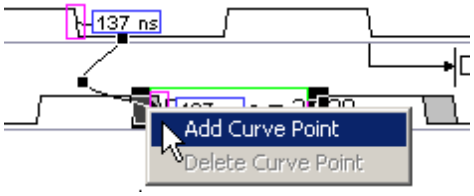
- Next, click on the parameter to select it and display the black curve points.



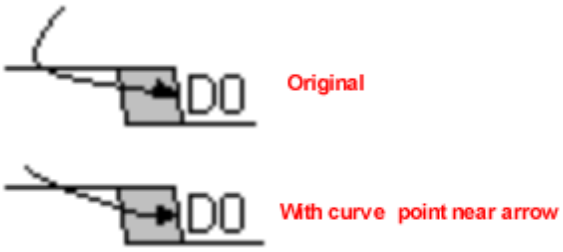
- You can change the curve by dragging and dropping a curve point.



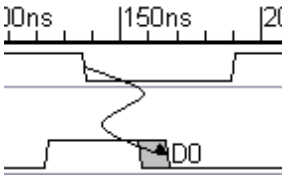
- You can add or delete curve points by ***right clicking*** on the curve and choosing the **Add Curve Point** or **Delete Curve Point** menu.



- If you add a curve point near the arrow head of the parameter, you can use it to control the angle of the arrow head.



- For our final edit, we moved both curve points so that the delay looks more flashy. We also returned **Display Label** to **Name** in the *Delay Properties* dialog.



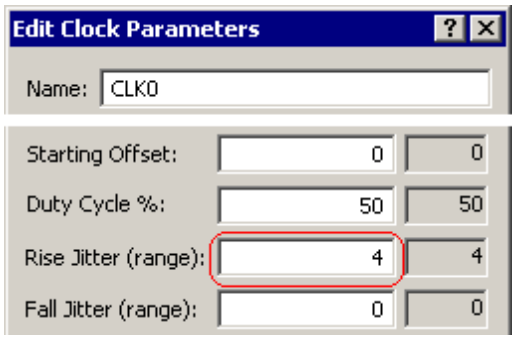
(TD) 3.7 Clock Jitter and Display

Clocks have many display and timing analysis settings that are covered in Timing Diagram Editor Manual Chapter 2: Clocks. All features that affect the timing analysis calculations for clocks are edited through the *Edit Clock Parameters* dialog. Features that only affect how the clock looks are reached through the *Signal Properties* dialog. In this section we will add edge jitter and see the effect on the distance measurement. We will also add arrows to the falling edge of the clock, change the slant of the waveform edges, and add grid lines to the clock.

Copyright © 2014, SynaptiCAD Sales, Inc.

Add edge Jitter with the Edit Clock Parameters dialog:

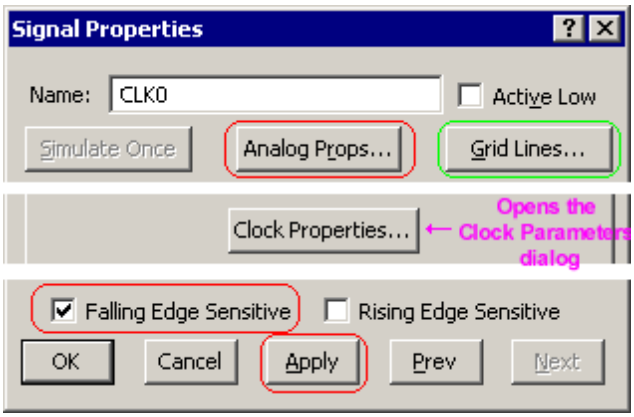
- Double click on waveform segment on CLK0 to open the *Edit Clock Parameters* dialog.
- Type **4** into the **Rise Jitter (range)** edit box and tab to another control, and press the **Ok** button to close the dialog.
- Notice the rising edges of the clock now show 4 ns of uncertainty.
- Also notice that the distance measurement shows the uncertainty with <48,52>, instead of the <50,50> display.





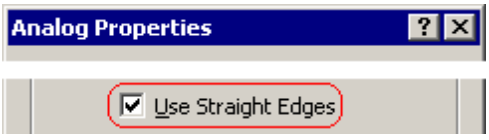
Add arrows, straight edges, and grid lines using the Signal Properties dialog

- Double click on the **CLK0** signal name to open the *Signal Properties* dialog.
- Check the **Falling Edge Sensitive** box and push the **Apply** button. This causes arrows to be added to the falling edge of the clock.





- Next, press the **Analog Props** button to open the *Analog Properties* dialog.

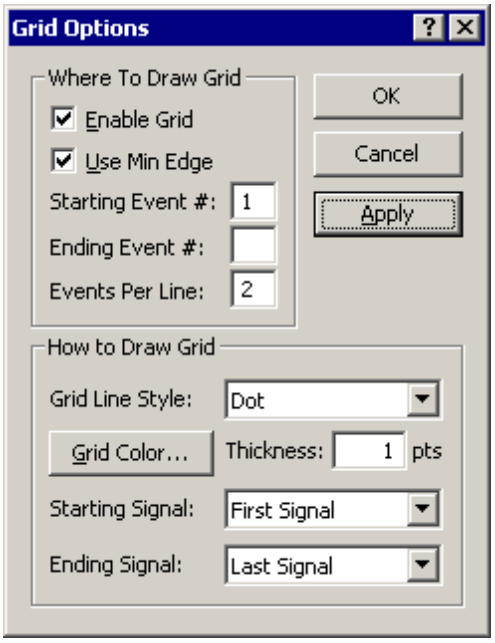


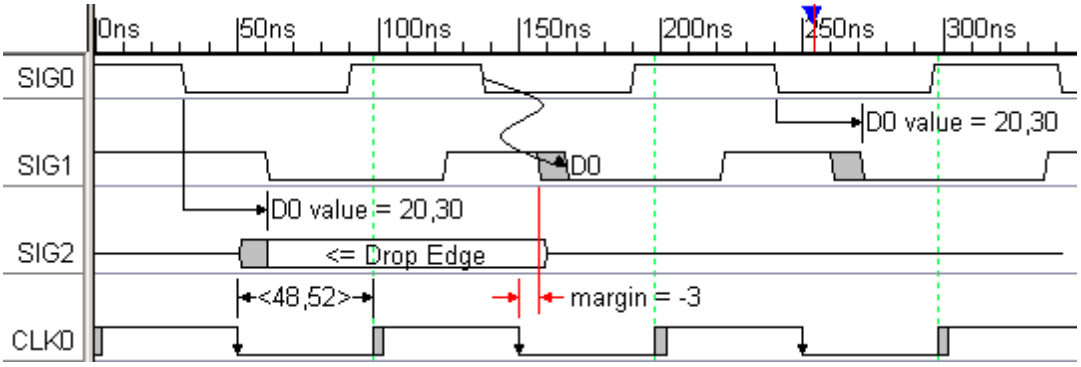
- Check the **Use Straight Edges** box and press **OK** to close the analog dialog. This will cause the clock to be drawn with straight edges instead of the normal slanted edges.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Next, press the **Grid Lines** button to open the *Grid Options* dialog.
- Check the **Enable Grid** box and press the **Apply** button. This draws grid lines on the clock.
- Play around with the grid options and make the grid draw on different edges. Also draw different color grids and line styles.



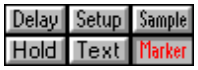


(TD) 3.8 Marker Time Compression

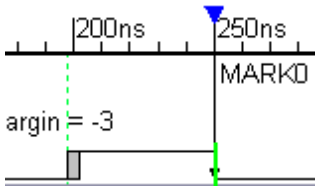
Time markers (vertical lines) can be added to a timing diagram for documentation, time compression, and to indicate the end of the diagram.

Add a Time Compression Marker.

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

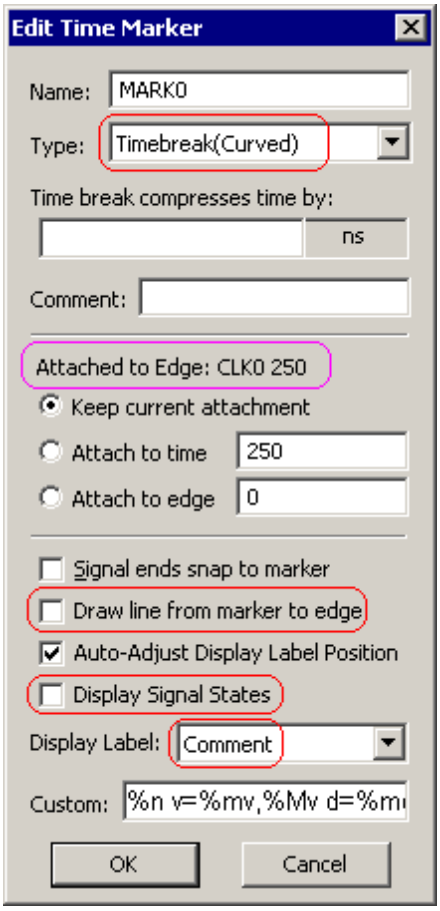


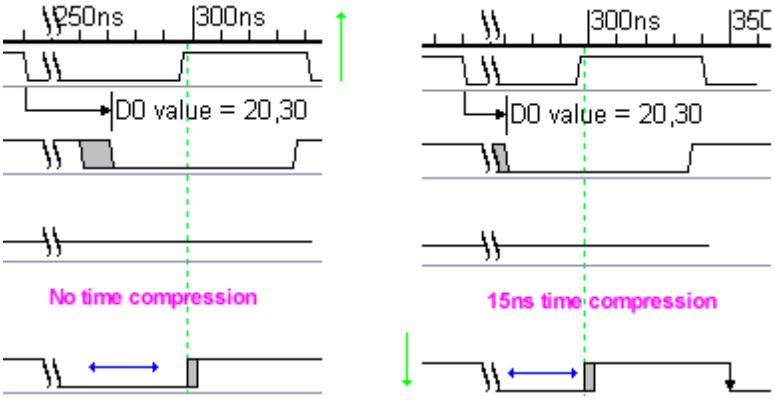
- ***Left click*** on the third falling edge of CLK0 (250ns), to select it, and then ***right click*** to add a Marker attached to that edge.



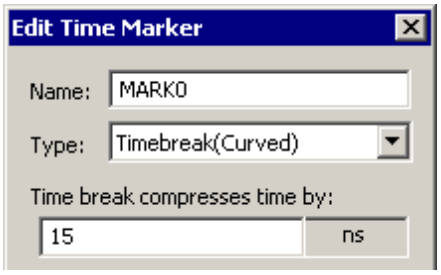
Copyright © 2014, SynaptiCAD Sales, Inc.

- Double click on the marker to open the *Edit Time Marker* dialog.
- Choose **Timebreak(Curved)** from the **Type** box to make the marker use a double curved line display.
- The **Attached to** display shows whether the marker is attached to a time or an edge. Since an edge was selected when you added the marker it was automatically attached to the selected edge.
- Uncheck the **Draw line from marker to edge** box. When marker is attached to an edge, this box determines if a dotted line will be drawn between the edge and the marker.
- Uncheck the **Display Signal States** so that the marker does not try to display the waveform values of the signals that are underneath it.
- In the **Display Label** box, choose **Comment**. Since the comment for the marker is blank, no label will be displayed for the marker.
- Press **OK** to close the dialog. Notice that the marker is curved and does not display its label. See picture below on the left





- Double click on the marker to open the *Edit Time Marker* dialog again.
- Type **15** into the **Time Break compresses time by** box and press **OK** to close the dialog.



- Notice in the above picture that 15ns of the next clock cycle is not displayed in the diagram. All

Copyright © 2014, SynaptiCAD Sales, Inc.

the parameters inside a compressed region continue to function,but part of the diagram is not shown.

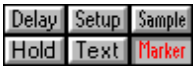
- Drag and Drop the Marker around the diagram and watch objects disappear and reappear.

(TD) 3.9 Marker Snap to Edge

A marker can also be used to indicate the end of a timing diagram. This is a useful feature if you are exporting to test bench formats. You can also make the ends of all the signals snap to the marker for a cleaner looking timing diagram.

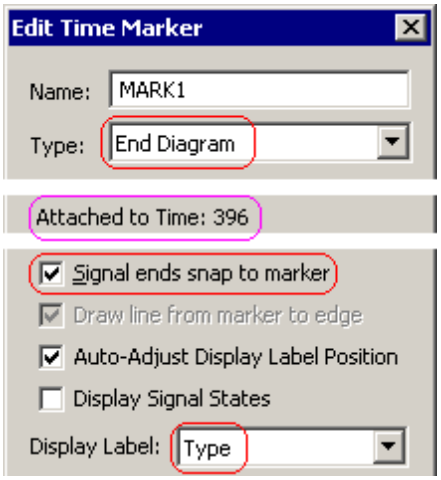
Add an End Diagram Marker with Snap to Marker

- In the last step we pressed the **Marker** button so that right clicks will add markers to the diagram.

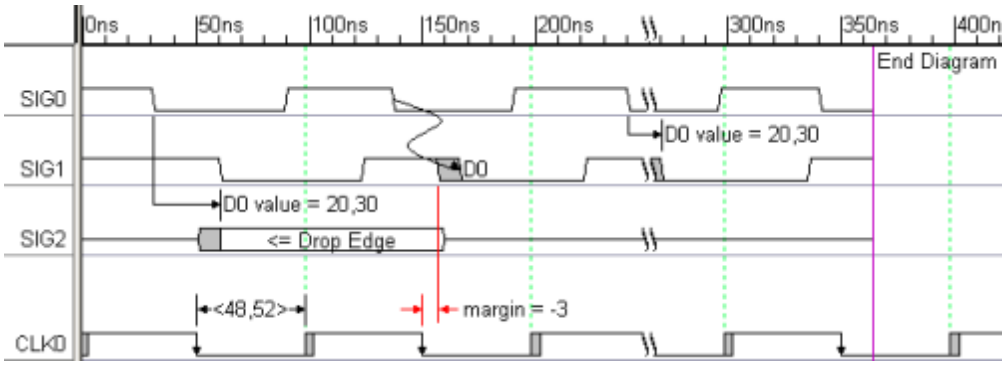


- Make sure that no edges are selected in the diagram, and then ***right click*** at the top of the diagram at about 400ns. This will add a marker to the right of all the drawn signals.

- Double click on the marker to open the *Edit Time Marker* dialog. Notice that the attachment is listed as **Time** because no edges where selected when the marker was added.
- From the Marker **Type** box, choose **End Diagram**. This causes the marker to draw itself with the purple simulation line.
- From the **Display Label** box, choose **Type** to make the marker display *End Diagram* as the display label (instead of its name).
- Check the **Signal ends snap to marker** box and press **OK** to close the dialog. Notice that all of the drawn waveforms have drawn themselves over to the marker.



- Drag and drop the end diagram marker and notice how the waveforms draw themselves.



Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 3.10 Marker Loops and Pipelines

Loop and pipeline markers can be used to document sections of the timing diagram.TestBencher Pro uses these markers to generate code for test benches and the code generation features are covered in the TestBencher Pro Basic Tutorial 142. Since these are markers will generate code, the timing diagram editor requires that both the beginning and ending markers are either attached to time or attached to the same type of edge on the diagram.

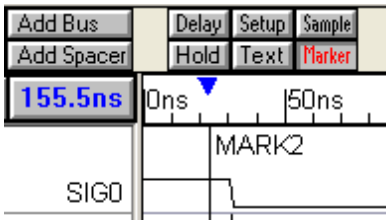
Add a Pipeline Marker:

Pipelines begin with a pipeline boundary marker and end at either another pipeline boundary marker or an end diagram marker. Since we already have an end diagram marker we will only need to add a beginning marker to draw the pipe.

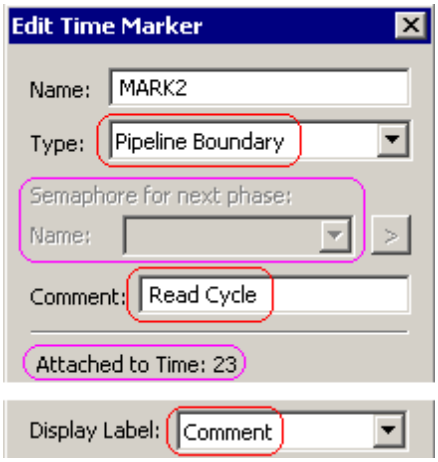
- In the last step we pressed the **Marker** button so that right clicks will add markers to the diagram.



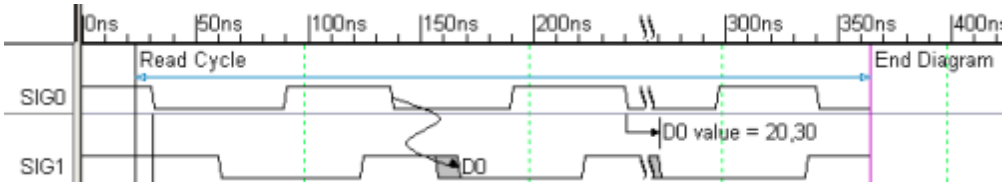
- Make sure that no edges are selected in the diagram, and then ***right click*** at the top of the diagram at about 25ns. This will add a marker to the right of all the drawn signals.



- Double click on the marker to open the *Edit Time Marker* dialog. Notice that the attachment is listed as **Time** because no edges were selected when the marker was added.
- From the Marker **Type** box, choose **Pipeline Boundary**. If the program can match this with another boundary marker or end diagram marker it will draw a blue horizontal line between the two markers.
- For most of the timing diagram editors, the **Semaphore** box is disabled. In TestBencher the Semaphore name would be centered above the pipe line marker.



- In the **Comment** box enter some text like **Read Cycle**.
- From the **Display Label** box, choose **Comment** to make the marker display its comment instead of its name.
- Press **OK** to close the dialog. Notice that a blue pipeline marker has been drawn across the diagram.

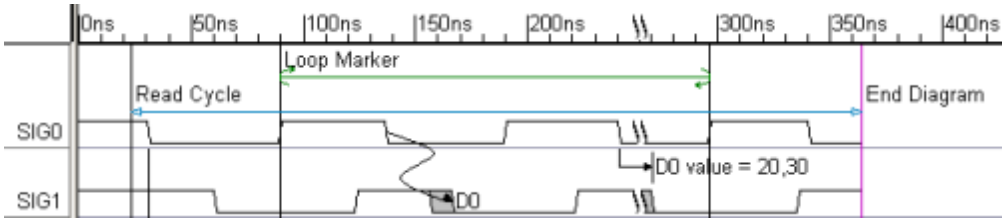


Copyright © 2014, SynaptiCAD Sales, Inc.

- The horizontal line can be moved by dragging and dropping the marker label to a new position.

Investigate Loop Markers:

Loops begin with one of the Loop marker types (**for loop**, **while loop**, **repeat loop**) and end on an **end loop** marker. Both the begin and end loop markers must be either both be attached to time or both be attached to the same type of edge on a particular signal. Since these are so similar to pipelines we will not add one in this tutorial, but the following picture shows what the Loop looks like in the diagram. This is a For loop attached to the rising edges of SIG0.

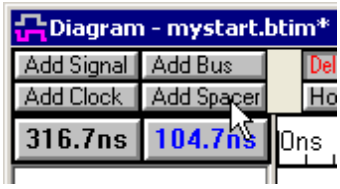


(TD) 3.11 Spacers and Text Font Controls

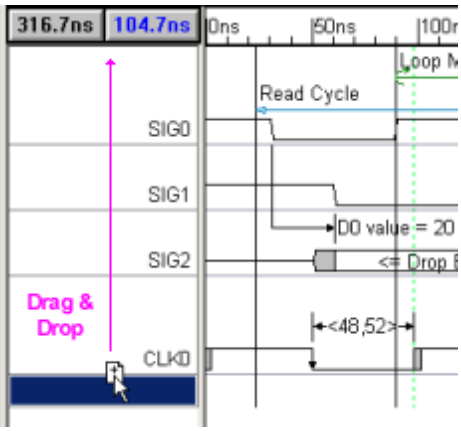
Text objects can be placed anywhere in a diagram to annotate cycles, edges, or segments. The font and color of each text object can be changed to stress the importance of that particular text object. The fonts also support superscripts, subscripts, and bold and italic attributes so your timing diagrams can use the same names and comments that are commonly used in data books. In this section we will add a Spacer signal to the top of the timing diagram and then add a title using a text object.

Add a Spacer and drag it to the top of the diagram:

- Press the **Add Spacer** button to insert a spacer into the diagram. If no signals are selected the spacer will be added to the bottom of the diagram.

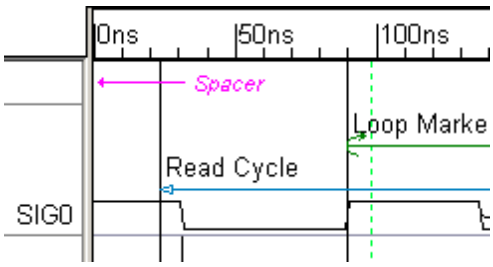


- Select the new spacer by left clicking on it in the label windwo.
- Then left click down on the selected spacer so that a paper icon appears and then drag the icon to the top of the label window and drop it.



Copyright © 2014, SynaptiCAD Sales, Inc.

- The Spacer adds a little space to the top of the timing diagram so that we will have room to place a title.

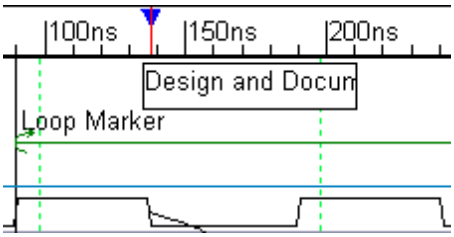


Add a Text Object to the Diagram and drag it to the top of the diagram:

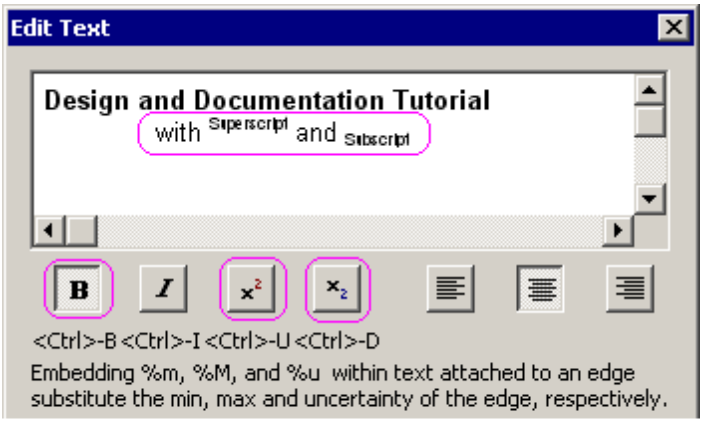
- Press the **Text** button so that right clicks will add text objects to the diagram.



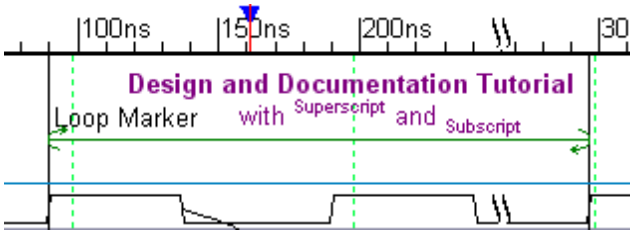
- At the top of the diagram, around 140ns, right click to open a text editing box and type **Design and Documentation Tutorial**, and then press the **Enter** key to close the editing box. This will add a text block to the top of the diagram using the default font.



- Double click on the text object to open the *Edit Text* dialog box.
- Add a second line of text that says **with Superscript and Subscript**.
- Select the first line of text and press the ***Bold button***.
- Select the **Subscripts** word and press the ***subscript button***.
- Select the **Superscripts** word and press the ***superscript button***.
- Press the **Font** button and change the color to purple then close the font dialog.



- Press Ok to close the text dialog
- Click on the text object and drag and drop it to a good location.



- Experiment by adding more text blocks.

Copyright © 2014, SynaptiCAD Sales, Inc.

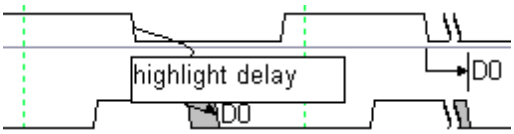
(TD) 3.12 Highlight Regions with Text Objects

Text Objects can also be used to highlight different regions of the timing diagram.

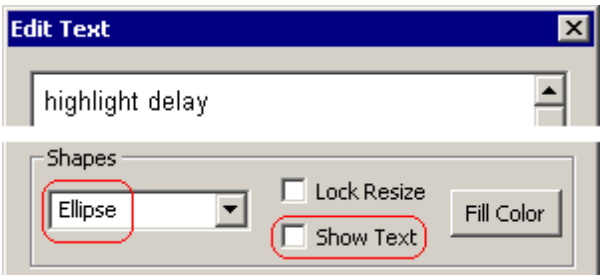
- Press the **Text** button so that right clicks will add text objects to the diagram.



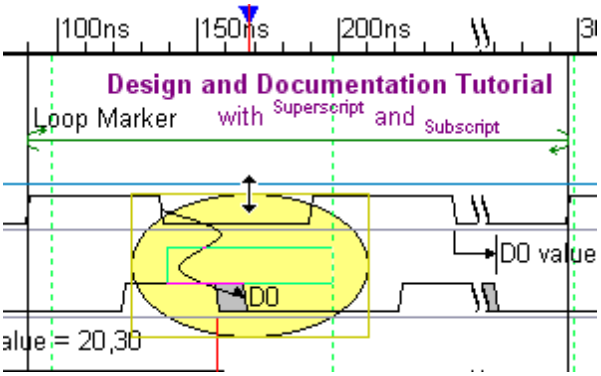
- **Right click** on top of the curved delay in the center of the diagram and type **Highlight delay** into the edit box.



- Double click on the text object to open the *Edit Text* dialog box.
- Choose **Ellipse** from the **Shapes** section.
- Uncheck **Show text** so that only the highlighted region will be displayed. The text will be used as a tool tip when the mouse goes over the region.



- Press **Ok** to close the dialog and display the highlighted region.
- Put the mouse over the top edge of the region so that it changes to an up-down arrow, then drag and drop the edge to resize the region to cover the entire delay.

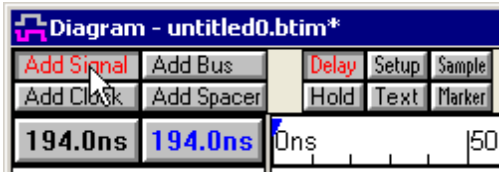


(TD) 3.13 Text and Hidden Signals

Text objects can be attached to edges of signals and centered inside segments so that as the timing diagram is scrolled or zoomed the text keeps its relative placement to the parent signal. One document trick that you can do is to attach a text object to a segment in a signal and then hide the signal so that only the text object is shown. Below is an example where we wish to label the cycles on the clock signal, but we do not want to display the documentation signal.

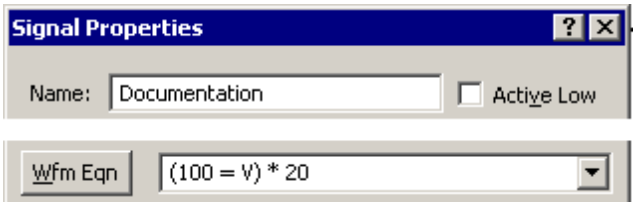
Add a Signal and Text Objects:

- Press the **Add Signal** button to add a signal to the bottom of the timing diagram.
- Double click on the new signal to open the *Signal Properties* dialog.

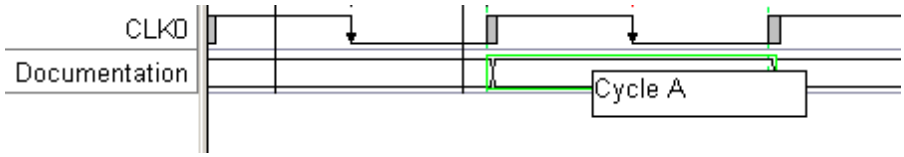


Copyright © 2014, SynaptiCAD Sales, Inc.

- Change the Name to **Documentation**.
- Enter **(100 = V)*20** into the waveform equation box and press the **Wfm Eqn** button to draw 20 valid segments that are 100 ns long (the period of the clock).



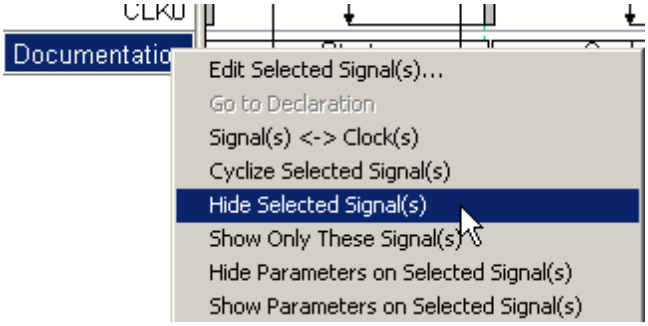
- Select a segment on the waveform and then right click to add a text object. Type in some text to document the clock cycle. Do this for several segments on the Documentation signal.



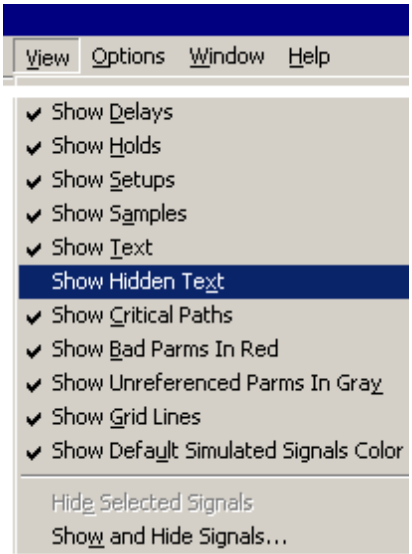
- Drag and Drop the text objects until they are at the vertical height that you want.



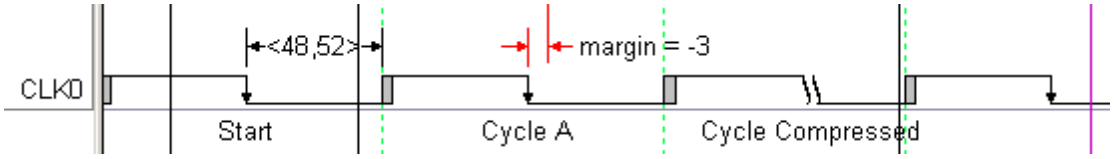
- Right click on the Documentation signal name and choose **Hide Selected Signal** from the context menu. This will hide both the signal and the text.



- Check the **View > Show Hidden Text** menu on the main program bar.
- Notice the **Show and Hide Signals** in the menu. This is the option that you use to show hidden signals.
- After you check Show Hidden test the text will appear, but the documentation waveform will not be visible.

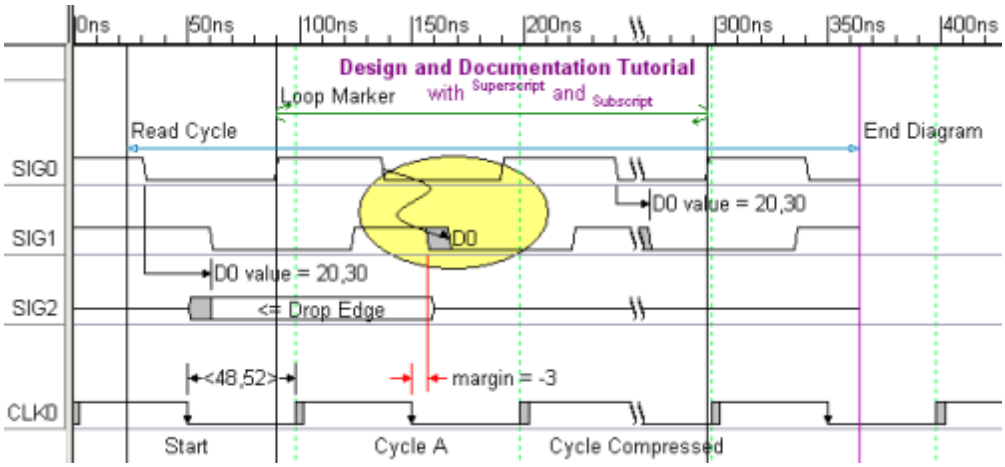


Copyright © 2014, SynaptiCAD Sales, Inc.



(TD) 3.14 Summary of Display and Documentation Tutorial

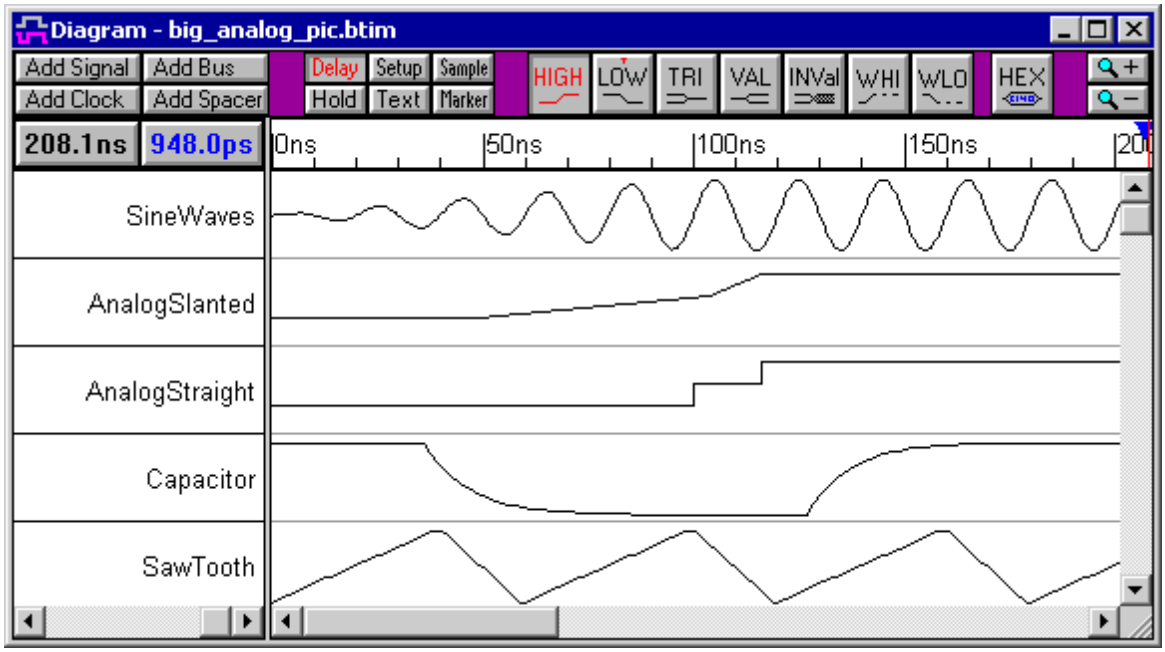
Congratulations! You have completed the *Display and Documentation* tutorial. In this tutorial you experimented with parameter display settings including how to add distance measurements and custom display strings. You have also touched on the some of the display options for markers, text objects, and clocks, but these objects have many more features that are covered in the manual.



Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 4: Analog Signals

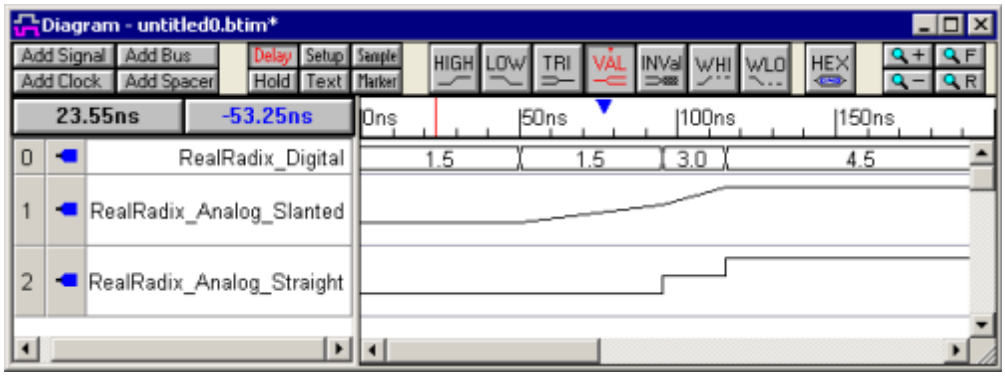
This tutorial helps you experiment with viewing, generating, exporting, and converting data values for Analog signals. Working through the tutorial will help you understand the different ways that waveforms can be manipulated so that you can create the exact waveforms needed for your designs.



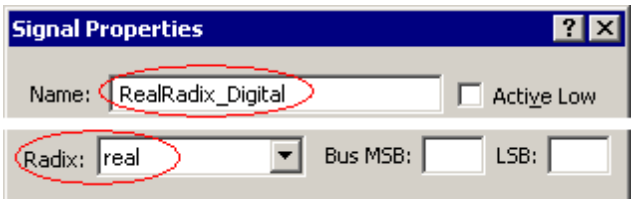
(TD) 4.1 Viewing Analog Waveforms

In this section you will draw the timing diagram shown below. Each of the three signals have the same waveform values, but are displayed using different settings. The first analog signal draws the waveform from point to point in a piecewise-linear way. The second analog signal shows the waveform drawn as step voltages.

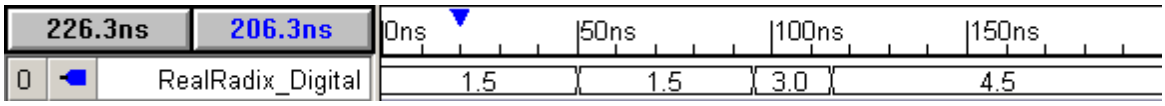
Copyright © 2014, SynaptiCAD Sales, Inc.



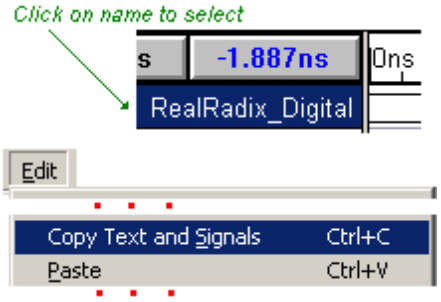
Add Two Digital Signals that are exactly the same:

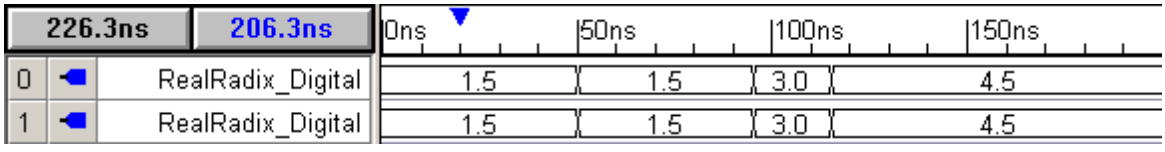


- Add one signal by pressing the **Add Signal** button.
- Double click on the signal to open the *Signal Properties* dialog.
- Change the name to **RealRadix_Digital**and choose the **real** radix from the radix box.
- Close the dialog.
- Next, sketch four valid segments as shown.
- Then double click on the first segment to open the *Edit Bus State* dialog and enter a **Virtual** state of 1.5. Use the **Next** button in the dialog to jump to subsequent segments and change the values to 1.5, 3.0, and 4.5. Close the dialog when you are done.



- Copy the completed digital signal by selecting the signal name then choosing the **Edit > Copy Text and Signals** menu. Then Paste it using the **CRTL-V** keys so that you have two identical signals.

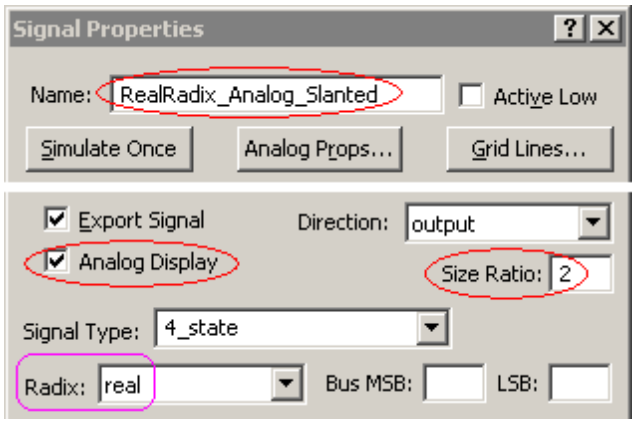


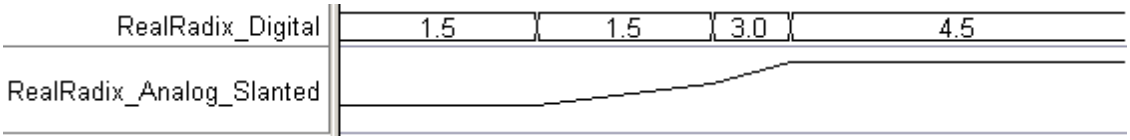


Copyright © 2014, SynaptiCAD Sales, Inc.

Setup the Analog Display for the default Slanted Display:

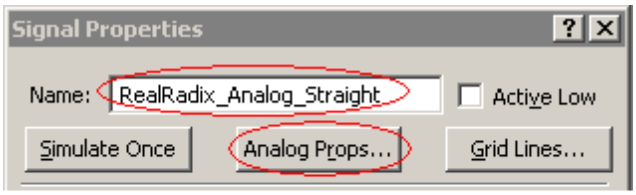
- Double click on the new signal to open the *Signals Properties* dialog and change its name to **RealRadix_Analog_Slanted** and check the **Analog Display** box so that the signal will display as a magnitude plot.
- By changing the **Size Ratio** to 2 or larger, the signal will be drawn taller so that it is easier to see the waveform.
- Close the dialog to display the signal as an analog magnitude plot.

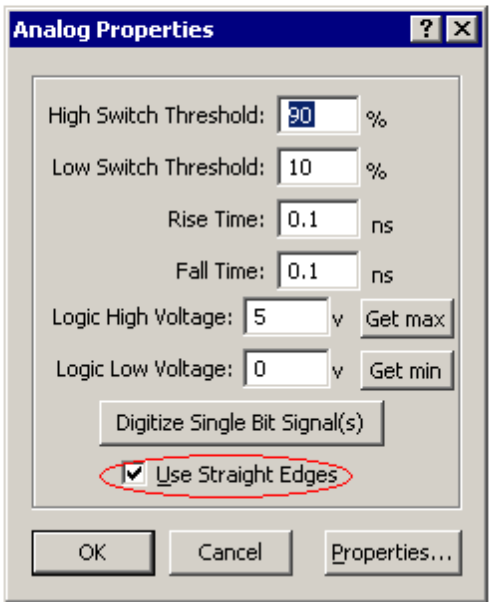




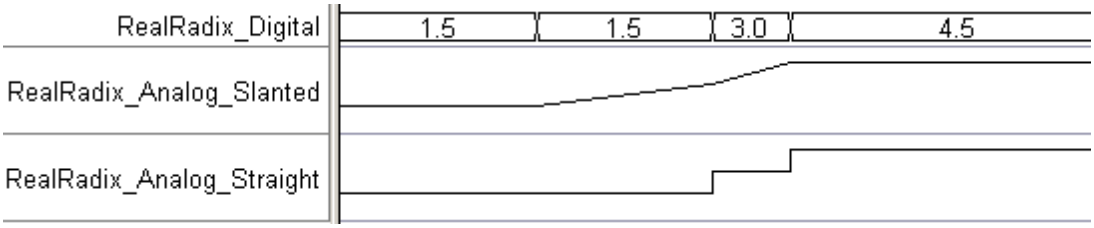
Display the signal as step voltages:

- Copy and Paste the **RealRadix_Analog_Slanted** signal by selecting the name and using the Ctrl-C and Ctrl-V keys.
- Double click on the new signal to open the *Signals Properties* dialog and change its name to **RealRadix_Analog_Straight**.
- Since this is a copy of the other analog signal, the **Analog Display** is checked and the **Radix** is set to ***real***.
- Next press the **Analog Props** button to open the *Analog Properties* dialog.
- Check the **Use Straight Edges** box to change the magnitude display. Press Ok to close this dialog.
- Notice when the steps occur in relation to the RealRadix_Digitial signal.





Copyright © 2014, SynaptiCAD Sales, Inc.



(TD) 4.2 Faster Drawing with Waveform Equation Blocks

The fastest way to generate analog waveforms that can be edited as equations is to use a Waveform Equation Block. Waveform equation blocks contain a Python function that is evaluated at each point to calculate the associated waveform value. There are several built-in functions and you can write your own functions.

Adding a Waveform Equation Block:

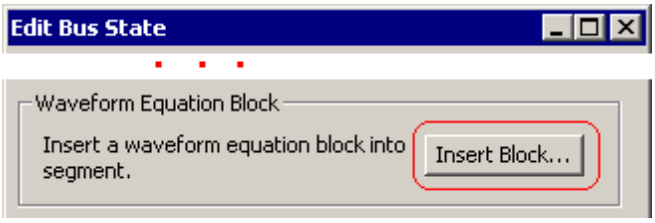
- Add a new signal and draw one or more waveform segments on it.



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

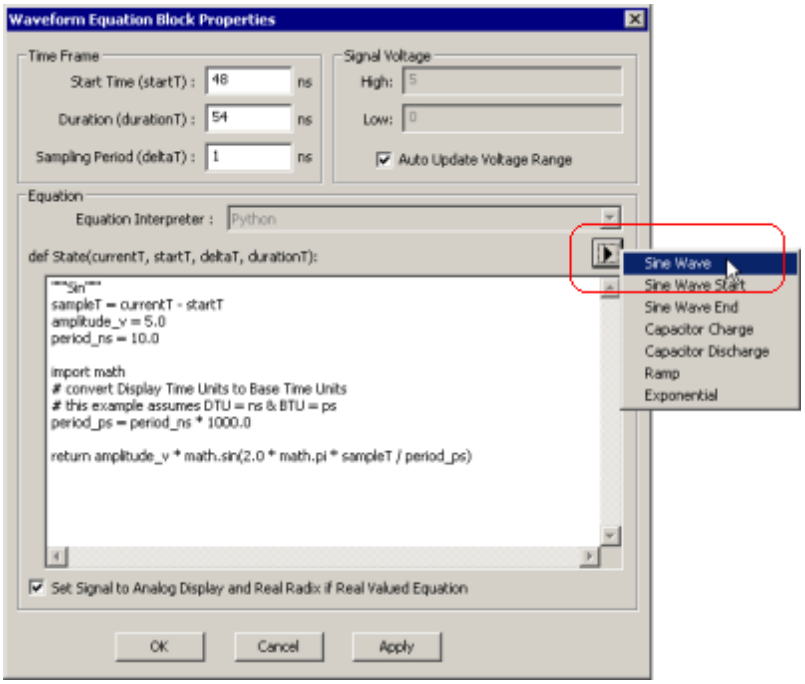


- Press the **Insert Block** button to open the *Waveform Equation Block Properties* dialog.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Choose **Sine Wave** from the pre-defined equation flyout.
- Notice that the code to make the sine wave is shown in the edit box portion of the dialog. This is Python code that can be edited by you.
- The **State** function is called at each point in the Sampling Period (**startT + n*deltaT**) to figure out the value for the waveform at that particular point (**n**).
- Press OK to close this dialog and generate the waveform.

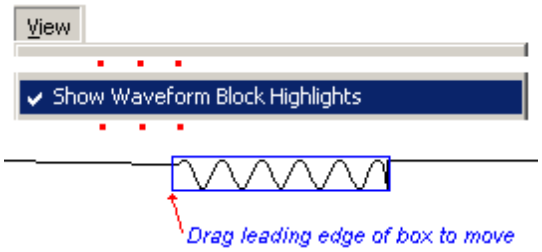




Setup View Mode and Drag the Equation block:

When you are viewing the waveforms normally, it is impossible to tell the difference between a sine wave generated by a Waveform Equation Block or one generated by the older method of State Label Equations. There is a special mode that will highlight the equation blocks and speed up editing these blocks (in this mode, double clicking on the block will directly open the dialog that edits waveform blocks).

- Check the **View > Show Waveform Block Highlights** menu to draw blue boxes around all of the Waveform Equation Blocks.
- Move the block around by dragging-and-dropping the first edge of the block using the left mouse button. This moves the block but does not affect the other edges of the signal.



- To move all of the transitions on one signal, hold down the **<1>** and **<2>** number keys while dragging the equation block. Holding down just the <1> key moves all the edges to the left of the selected edge, and the <2> key moves all the edges to the right of the selected edge. You may need to add more transitions on the signal to see the effect of these keys.

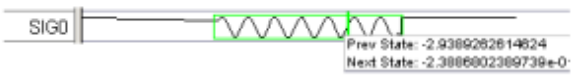
(TD) 4.3 Writing Python Waveform Equation Blocks

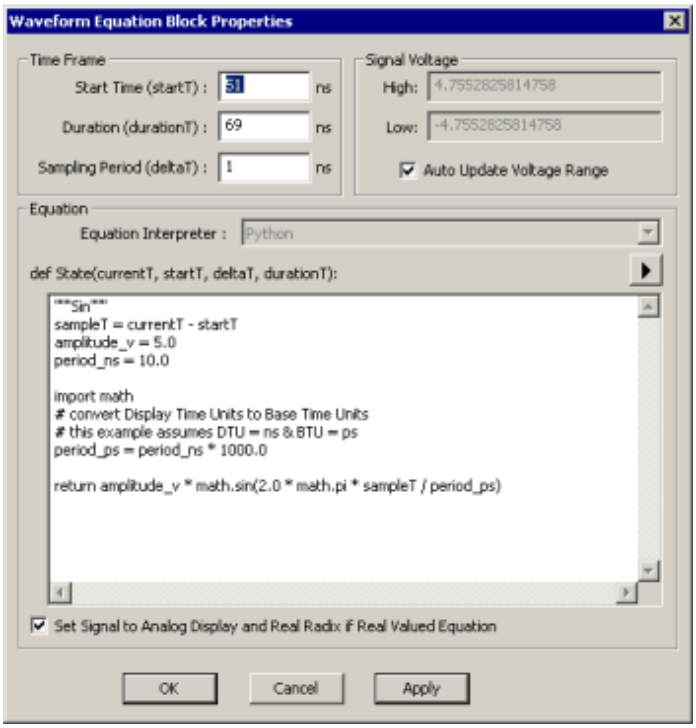
If you use the pre-defined waveform equation blocks, you will edit the code to control the different parameters like period or amplitude. You can also choose to write your own equations.

Copyright © 2014, SynaptiCAD Sales, Inc.

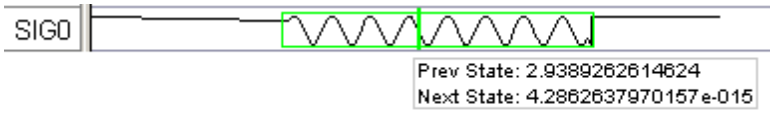
Edit the default Sine Wave:

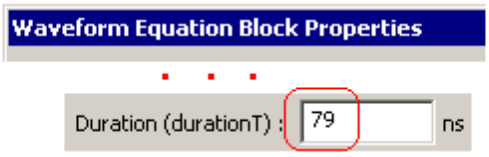
- Double click on the equation block to open the ***Waveform Block Properties*** dialog. If the *Edit Bus State* dialog opens instead, then press the **Edit Block** button to get to the ***Waveform Block Properties*** dialog.
- The **State** equation is called to calculate the value for each point on the waveform.
- The points are at (**startT** + n * **deltaT**). It stops when **durationT** is reached.
- Changing the **Start Time** is the equivalent of dragging and dropping the leading edge of the block's selection box.
- Making the Sampling Period smaller will create more points in the block (e.g. durationT/ deltaT points will be created by the block).





- Make the sine wave cover a longer period of time by adding 10ns to the **Duration** and pressing the **Apply** button.





Investigate the Code:

- The first line is a comment string that defines the name of the function. It does not have to be included.

```
"""Sin"""
```

- Here sampleT is defined as a relative time from the start of the block so that the waveform shape is independent of the start time of the block.

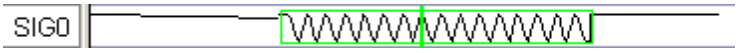
```
sampleT = currentT - startT
```

- These lines will normally be edited to control the period and amplitude of the signal.

```
amplitude_v = 5.0
period_ns = 10.0
```

- Change the period to **5.0** and press the **Apply** button to see the period change.

Copyright © 2014, SynaptiCAD Sales, Inc.



- This imports the standard Python math library so you do not have to define what basic math functions like sin do.

```
import math
```

- You must keep track of the time units that you are working on.

```
# 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)
```

- In the last line, a real value is returned for the current sample point. The **math.sin( )** function is the sine function in the Python **math** library.

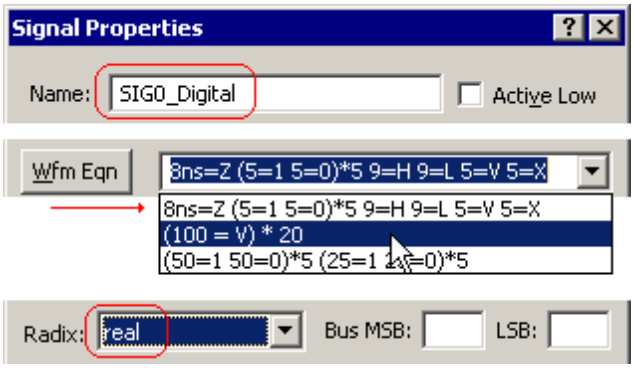
(TD) 4.4 State Label Equation Alternative

Another way to generate analog signals is to use Waveform and Label equations. This is a little simpler than writing the Python code required when using the Waveform Equation Blocks of the previous section. However, once the waveform is generated you cannot edit it using the original equation. You will have to either edit each edge, or delete a section of the waveform and regenerate it.

In this example, we will use a Waveform Equation (not a block equation) to generate the blank waveform segments. Then use a State Label equation to insert the values onto the segments. We will be inserting an extra segment for each value change so that we can use it in the next section to render a "stepped" waveform even though we have not checked the **Use Straight Edges** checkbox.

Use a Waveform Equation to Generate the Valid segments:

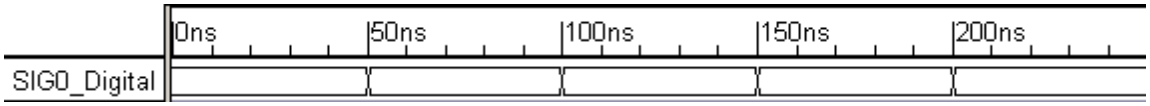
- Add a new signal and double click to open the *Signals Properties* dialog. Name the signal **SIG0_Digital** and set the **radix** to ***real***.



- Notice the default equation in the **Wfm Eqn** box. It is an example of all the syntax accepted by the generator. Each segment is represented by a **time=state** pair. If you forget the syntax just generate this equation and figure it out from the waveform.
- Enter **(50=V)*5** to generate 5 valid segments that are 50ns long.



- Press the **Wfm Eqn** to generate the waveform segments. Leave the dialog open.



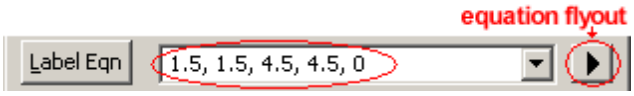
Use a simple Label Equation to label the segments:

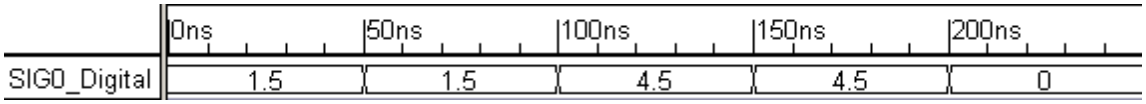
The Label Equation box accepts equations that generate state values for the signal's waveform segments. This example demonstrates the concatenation operator (the comma). The concatenation

Copyright © 2014, SynaptiCAD Sales, Inc.

operator can also be combined with the decrement and increment label equations to create more complex patterns.

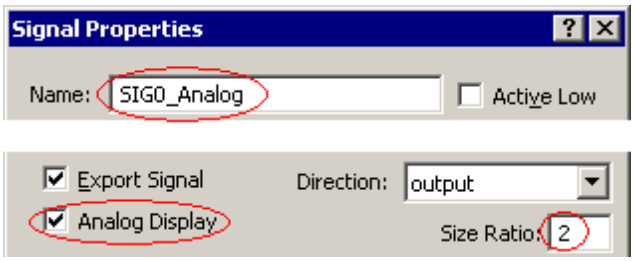
- In the **Label Eqn** box, type **1.5, 1.5, 4.5, 4.5, 0**.
- Press the **Label Eqn** button to apply the equation.

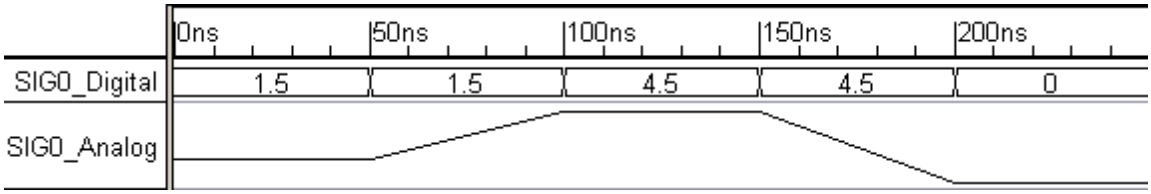




Create an Analog Signal:

- Copy and Paste SIG0_Digital by selecting the signal name and using the **Ctrl-C** and **Ctrl-V** keys. Double click on the new signal to open the *Signal Properties* dialog.
- Name the signal **SIG0_Analog**
- Check the **Analog Display** check box.
- Set the **Size** to **2**, to expand the waveform.
- Press the **Apply** button to apply the changes to the signal.





Although the above analog signal appears to have ramps in it, it will look like a stepped waveform when exported to a digital simulator. To create ramps that export to a digital simulator, use the **Ramp** label equation as discussed later in this tutorial.

(TD) 4.5 Drawing a Step Signal

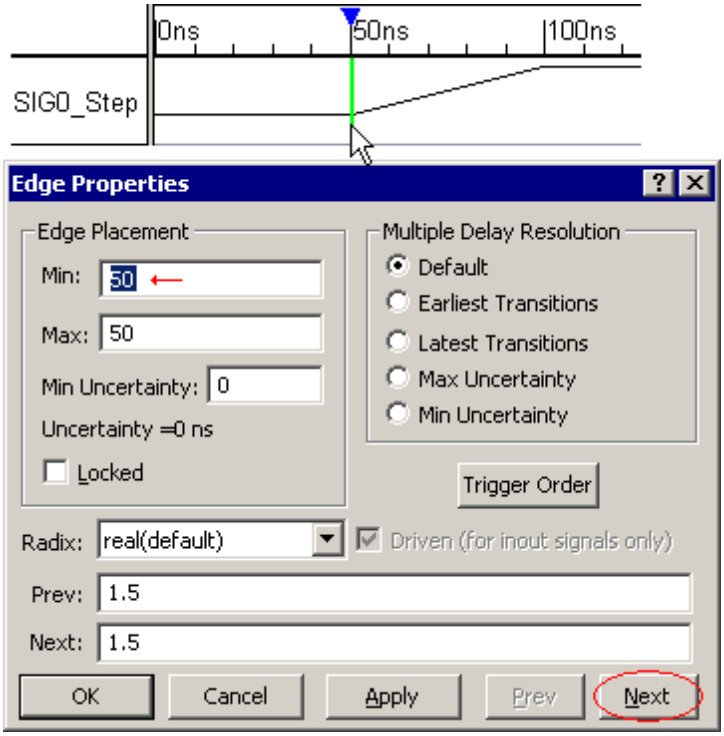
The analog signal from the previous step can now be easily converted to a step signal by adjusting the edge times for the segment that displays the change in value (The second and the fourth segments in our example). You can either use the mouse to move one of the edges of the segment to the same time as the other or you can use the *Edge Properties* dialog to adjust the edge times.

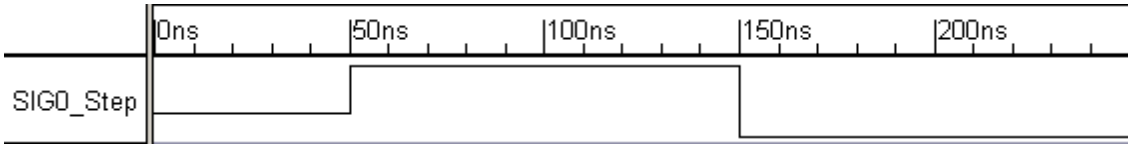
To use the Edge Properties dialog:

- Copy and Paste SIG0_Analog by selecting the signal name and using the **Ctrl-C** and **Ctrl-V** keys. Double click on the new signal to open the *Signal Properties* dialog and name the new signal **SIG0_Step**.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Double-click the first edge to open the edge properties dialog.
- Make note of the **Min** time (50ns).
- Click the **Next** button to move to the next edge (at 100ns).
- Enter the Min time from the previous edge (50ns) in either the **Min** or **Max** edit box. Click Next to move to the next edge and apply the changes.
- Notice that there is a vertical step at 50ns.
- Move the edge at 200ns to **150ns** using the same technique.





(TD) 4.6 Generating Sine Waves

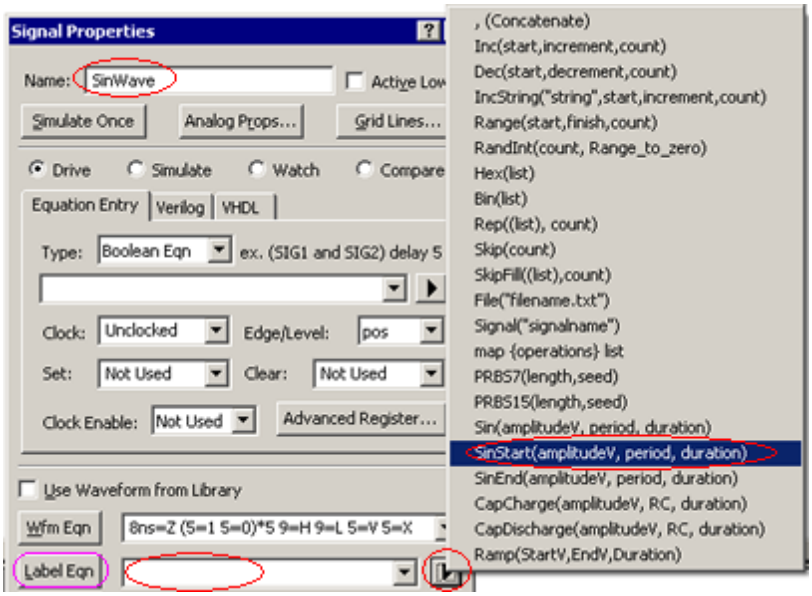
Both Waveform Block Equations and Label Equations can be used to generate Sine wave signals. There are three functions **SinStart**, **Sin**, and **SinEnd** that generate sine waves with a growing, steady, or decreasing amplitude. Both methods produce waveforms that look the same. However, Waveform Block Equations can be edited to tweak what the waveform looks like. With State Label Equations, each generated state is a separate event and must be individually edited or erased if changes are needed.

SinStart function with State Label Equations

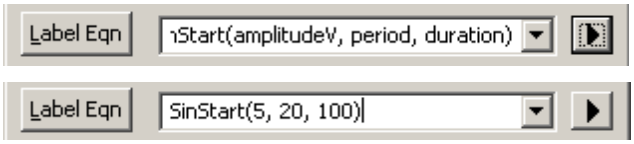
First we will start with the State Label equations where each function has parameters for **amplitude**, **period** and **duration**. There is also an optional fourth parameter that can specify the number of **points** to use when drawing the waveform. It is not necessary to memorize the syntax of the function, because the equation fly-out will tell you the parameters and function names. The SinStart functions starts a Sine wave with amplitude of 0 and grows to the amplitude specified over the duration specified.

Copyright © 2014, SynaptiCAD Sales, Inc.

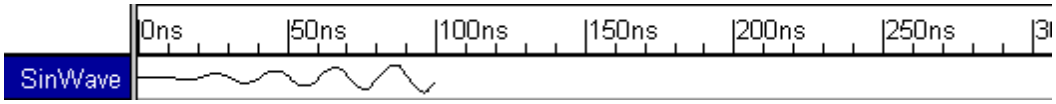
- Add a new signal and double click on it to open the *Signal Properties* dialog. Name the signal **SinWave**, then delete any text that might be in the Label Eqn box. Adjust the placement of the dialog so that you can see the waveform area and the dialog at the same time.
- Next click on the Label Eqn fly out button (see below) and choose **SinStart** from the list of available functions.



- Notice that the SinStart function has been added to the Label Eqn box.
- Edit the parameters so that signal will have amplitude of 5, a 20ns period and has reached its full amplitude by 100ns (assuming that ns is the display time unit)



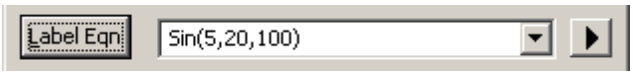
- Press the **Label Eqn** button to generate the start of a Sine wave. The timing diagram editor will automatically change the radix of the signal to **real**, because the generator uses floating point numbers to model the waveform. The waveform should look similar to this:



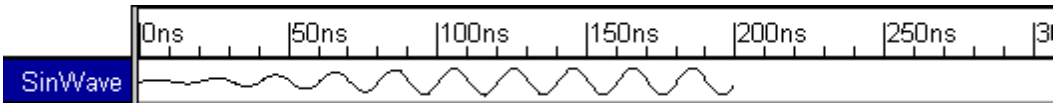
Sin function with State Label Equation

The Sin function draws a continuous Sine wave using the specified parameters. Here, we will append to the starting Sine wave drawn on the **SinWave** signal.

- Enter **Sin(5, 20, 100)** in the *Label Eqn* box either by typing or using the equation flyout.



- Press the **Label Eqn** button to continue the Sine wave generation.



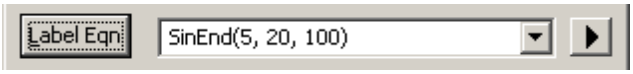
Notice that by using the same parameters, the generated Sine wave matches the Start Sine wave that was drawn.

Copyright © 2014, SynaptiCAD Sales, Inc.

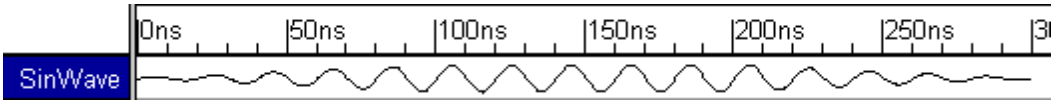
SinEnd function

The SinEnd function does the inverse of the SinStart function. It starts with a Sine wave of the full amplitude and diminishes over the specified duration until the amplitude is zero.

- Enter **SinEnd(5, 20, 100)** in the *Label Eqn* history either by typing or using the equation flyout.

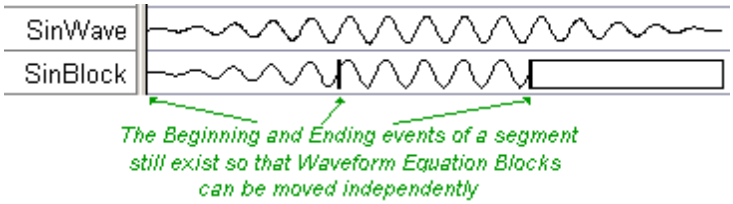


- Press the **Label Eqn** button to finish our Sine wave generation.



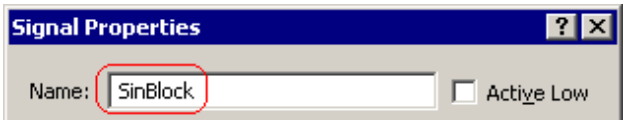
Sine waves with Waveform Block Equations

Waveform Equation Blocks preserve the beginning and ending edges of the block so that the block can be dragged and dropped independently of other blocks on the same signal. In the picture below, the waveform was created by inserting waveform equation blocks into consecutive valid segments. This caused unsightly black lines to be drawn between the segments.

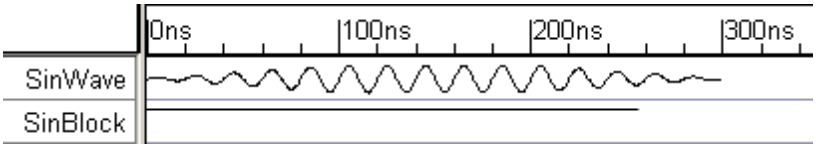


If you delete the black lines between two waveform equation blocks, then the blocks are said to be ***chained*** together. Each block can still be edited, but the starting time of the second block is now fixed at the ending time of the first block. A quick way to add chained blocks is to add a block to the middle of a long waveform segment:

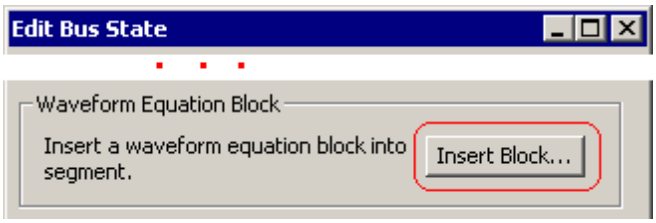
- Add a new signal and double click on it to open the *Signal Properties* dialog. Name the signal **SinBlock**.



- Draw one long segment on the **SinBlock** signal that is at least 250ns long.

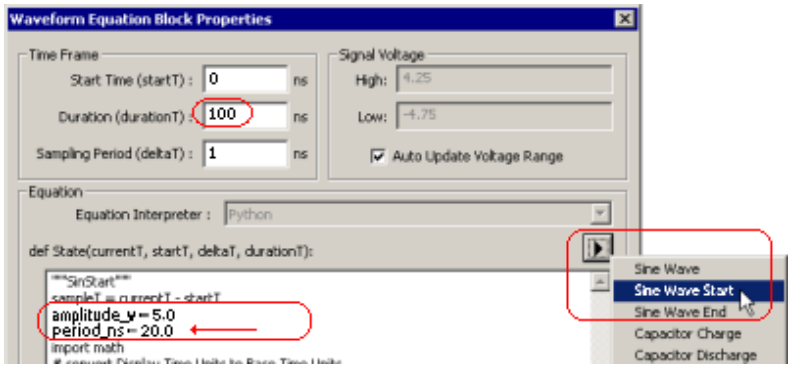


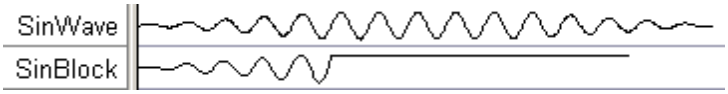
- Double click on the segment to open the *Edit Bus State* dialog.
- Press the **Insert Block** button to open the *Waveform Equation Block Properties* dialog.



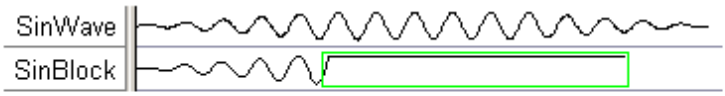
Copyright © 2014, SynaptiCAD Sales, Inc.

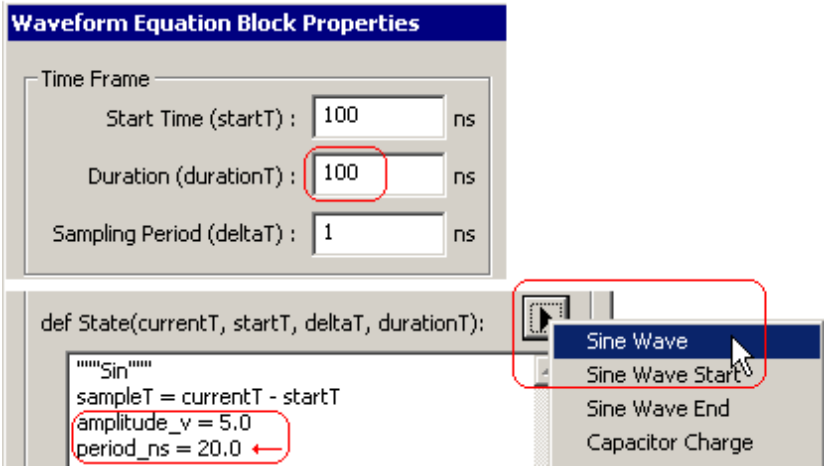
- Choose **Sine Wave Start** from the flyout.
- Change **Duration** to **100** ns.
- Edit the code so that **period_ns = 20.0**
- Press the **OK** button to close the dialog and draw the waveform. Notice that there is not a black line at the end.

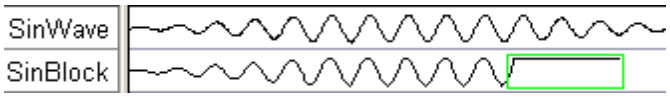




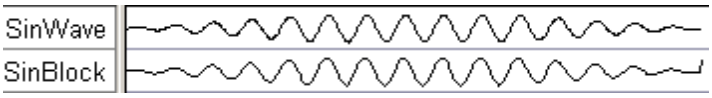
- Double click on rest of the segment to open the *Waveform Equation Block Properties* dialog.
- Choose **Sine Wave** from the flyout.
- Change **Duration** to **100** ns.
- Edit the code so that **period_ns = 20.0**
- Press the **OK** button to draw the waveform.







- Add the sine wave ending section by choosing **Sine Wave End** from the flyout and changing **Duration** to **100**ns and setting the **period_ns** to **20**.
- There is a small black line at the end of the waveform because this is the event that got moved from 250ns. You can delete it if desired.



(TD) 4.7 Generating Capacitor Charge and Discharge

Both Waveform Block Equations and Label Equations can be used to draw capacitor charge and discharge waveforms. These functions have three required parameters - **amplitude, RC constant,**

Copyright © 2014, SynaptiCAD Sales, Inc.

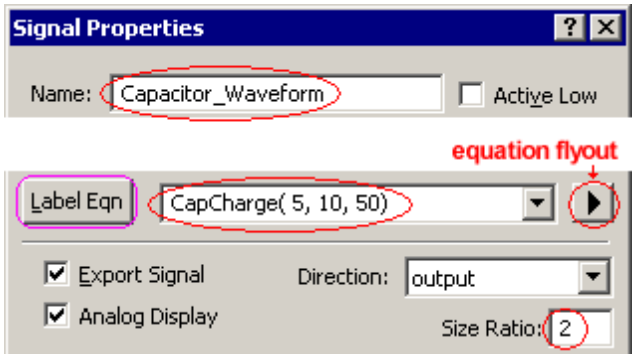
and **duration** - that are used to generate the waveform.

For Label equations, there is also an optional fourth parameter that can specify the number of **points** to use when drawing the waveform. For Waveform Block Equations, the number of points is determined by **durationT/deltaT**.

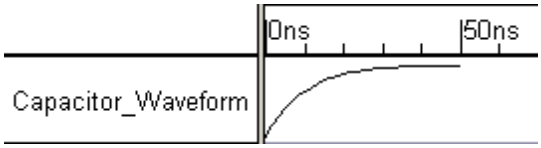
Capacitor Charging Waveforms with State Label Equations

The **CapCharge** function generates the waveform for a charging capacitor. The waveform will start at zero and gradually rise (using the RC value) to the amplitude at the end of duration specified.

- Add a new signal to the diagram and double click on it to open the *Signal Properties* dialog. Name the signal **Capacitor_Waveform** and set the Size ratio to 2.
- Enter **CapCharge(5, 10, 50)** in the *Label Eqn* box either by typing or by using the equation flyout. The parameters are amplitude of 5, RC constant of 10, and duration of 50ns (assuming the ns is the display time unit).



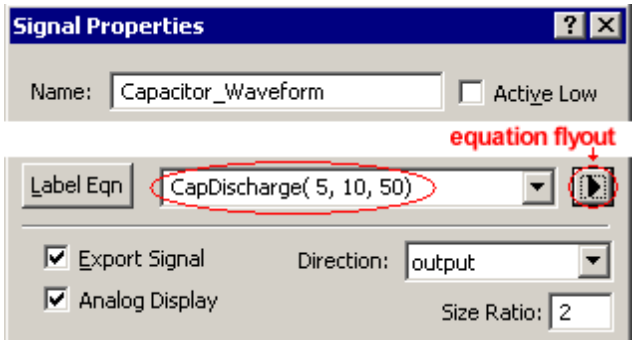
- Click **Label Eqn** to generate the capacitor charging waveform. The timing diagram editor will automatically change the radix of the signal to **real**, because the generator uses floating point numbers to model the waveform.



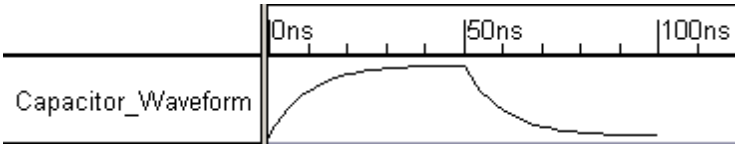
Capacitor Discharge Waveforms with State Label Equations

The **CapDischarge** function performs the inverse of the **CapCharge** function. The waveform starts at the maximum amplitude and it slowly declines based on the RC provided over the duration until it reaches the lowest point.

- Enter **CapDischarge(5,10,50)** in the *Label Eqn* box by typing or by using the equation flyout. This generates a capacitor discharge waveform that starts at an amplitude of 5, and discharges at a rate controled by an RC value of 10, and a duration of 50ns (assuming that the display time unit is ns).



- Press the **Label Eqn** button to generate the waveform.



The Capacitor label equations append to the end of the waveform, so you can also draw part of the

Copyright © 2014, SynaptiCAD Sales, Inc.

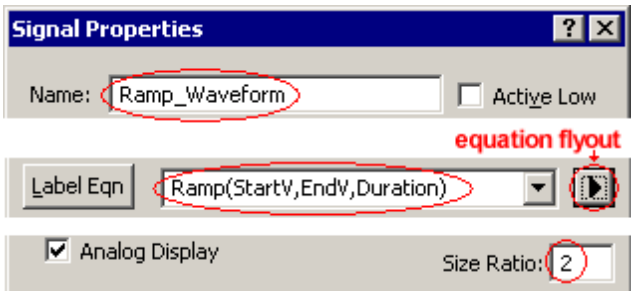
waveform and then append a capacitor waveform to the drawn signal.

(TD) 4.8 Generating Ramp Waveforms

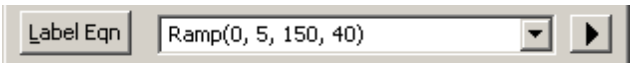
The best way to create a ramp signal is to either use a **Ramp** label equation or a Ramp waveform block equation. These functions can create a ramp signal with plenty of data points, so that it will export accurately to both analog and discrete-event digital simulators. The functions have three required parameters: **startVoltage, endVoltage,** and **duration.** For Label equations, there is also an optional fourth parameter that can specify the number of **points** to use when drawing the waveform.

Draw a ramp with a State Label Equations

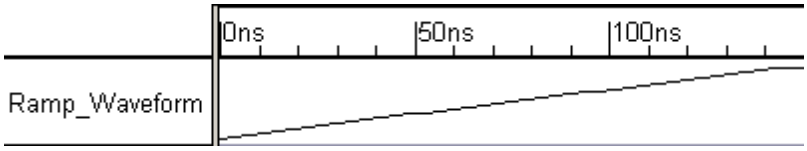
- Add a new signal to the diagram and double click on the signal name to open the *Signals Properties* dialog. Name the signal **Ramp_waveform** and set the Size ratio to **2**.
- Clear out any text that might be in the label equation box, then use the equation flyout button to insert a Ramp function.



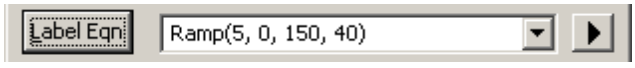
- Edit the parameters as shown to make a ramp that starts at 0 volts and ramps up to 5 volts, over a period of 150 ns. Also add the fourth parameter to specify that the ramp should be generated using 40 points.

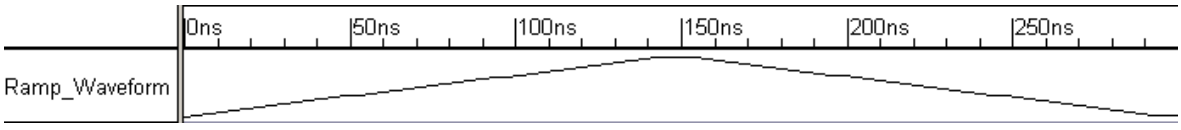


- Press the **Label Eqn** button to generate the waveform.



- Edit the parameters to make a down ramp, then press the **Label Eqn** button to generate the waveform.



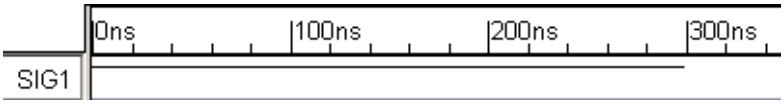


(TD) 4.9 Random Analog Equations

Waveform Block Equations can also be used to generate random digital and analog data.

Generate a Digital Signal with Random Values

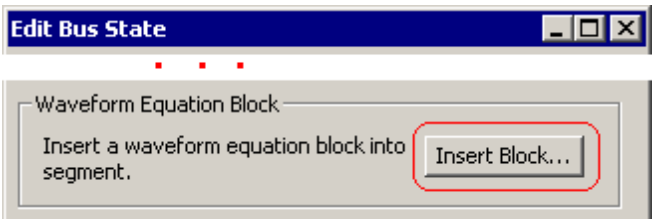
- Add a new signal and draw a waveform segment that is about



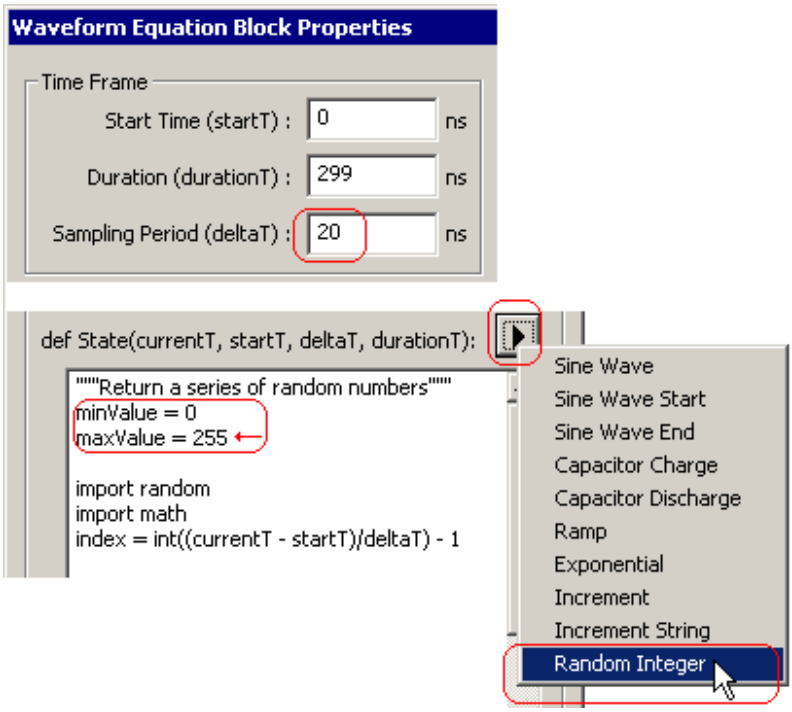
Copyright © 2014, SynaptiCAD Sales, Inc.

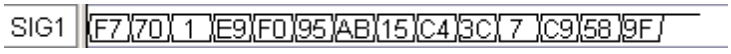
300ns long.

- Double click on the segment to open the *Edit Bus State* dialog.
- Press the **Insert Block** button to open the *Waveform Equation Block Properties* dialog.



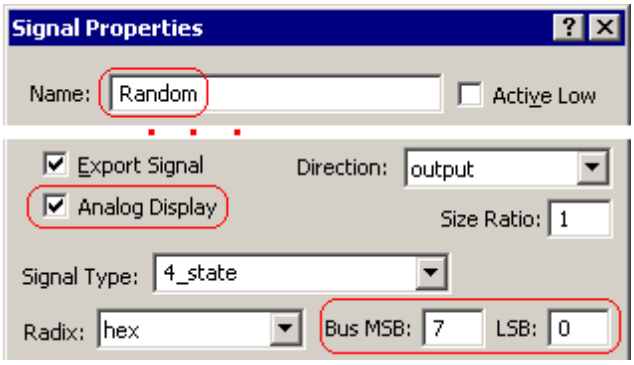
- Change the **Sampling Period** to **20ns**.
- Choose **Random Integer** from the pre-defined equation flyout.
- In the code block, change the **maxValue** to 255.
- Press the **Ok** button to close the dialog and generate the digital signal.
- Notice that the digital segments on the signal are 20ns long and the random values are between 255 and 0.

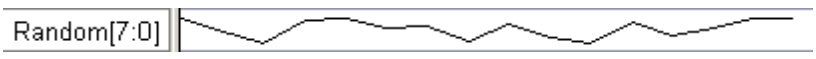




Display the Signal as Analog:

- Double click on the signal name to open the *Signal Properties* dialog.
- Change the name to **Random**.
- Check the **Analog Display** box.
- Change the **MSB** to **7** and the **LSB** to **0**. This will make it an 8 bit signal which can display the random values in the 255 to 0 range.
- Press the **Ok** button to close the dialog and display the signal.





Copyright © 2014, SynaptiCAD Sales, Inc.

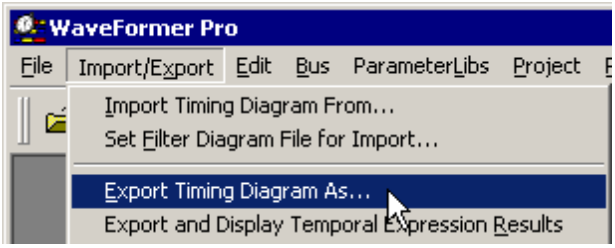
(TD) 4.10 Exporting to SPICE, VHDL, and Verilog

WaveFormer Pro can export waveforms to many different formats by using the **Import & Export > Export Timing Diagram** menu and saving to the appropriate format.

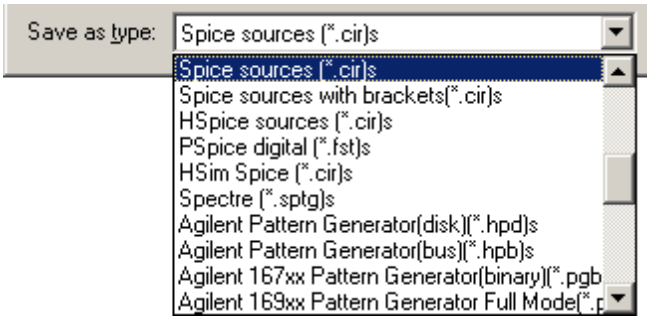
Exporting Analog Signals To SPICE:

When exporting an analog signal to SPICE, the **straight edges** box in a signal's *Analog Properties* dialog determines whether the signal is modeled as piecewise-linear or as step voltages. Step voltages are approximated by adding an additional point in the spice PWL statement immediately after each drawn point. Digital signals always export to SPICE as quasi step voltages using PWL statements. To export to SPICE:

- Choose the **Import/Export > Export Timing Diagram As** menu to open the *Export As* dialog.

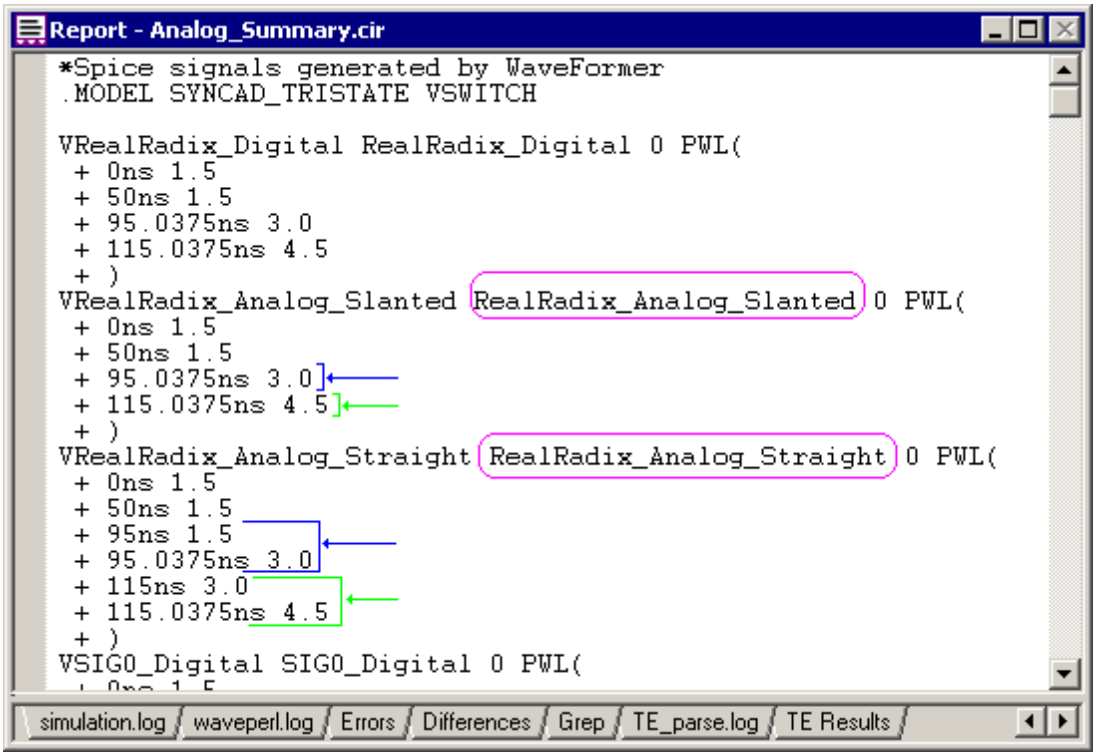


- Choose one of the Spice formats, like **Spice sources** from the **Save as type** box, then save the file. WaveFormer will save the file and also display it in the Report Window.



- Notice the differences between the code for the **RealRadix_Analog_Slanted** and **Real Radix_Analog_Straight** signals that were drawn in the first section. When the straight edges box is checked, there are extra points where the voltage changes abruptly in a short period of time to represent the step voltage changes.

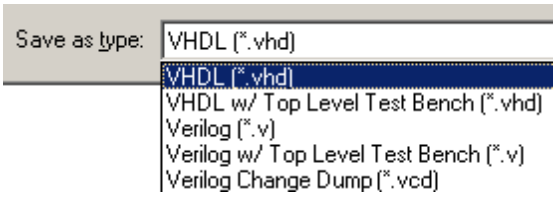
Copyright © 2014, SynaptiCAD Sales, Inc.



Exporting to VHDL or Verilog Simulators:

When exporting analog signals to a discrete event simulator such as VHDL or Verilog, analog signals must be exported as step voltages (discrete event simulators cannot model a true ramp, for example, so ramps must be approximated by step voltages). So regardless of whether an analog signal is being rendered piecewise-linear or as step voltages in the timing diagram window, it will export as step voltages to HDL simulators.

The export is performed as show above, except that one of the VHDL or Verilog formats is chosen in the Save As Type box,



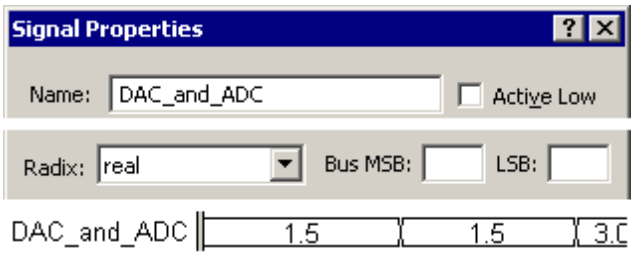
(TD) 4.11 ADC and DAC Conversion

When working with multi-bit signals, changing **radix** or the **msb/lsb** performs a conversion similar to an Analog-to-Digital or Digital-to-Analog converter, depending on the direction that you are going in. For single-bit signals, like a digital waveform captured by a digitizing oscilloscope or from a SPICE simulation of a digital circuit, you can convert the analog data into digital data with uncertainty at the edges by using the **Digitize Single Bit Signal** button in the *Analog Properties* dialog of the signal.

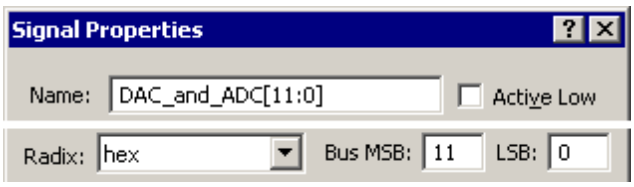
Copyright © 2014, SynaptiCAD Sales, Inc.

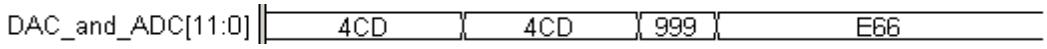
ADC conversion on a multi-bit signal:

- Copy and paste the signal **RealRadix_Digital** using the CTRL-C and CTRL-V keys (this is the signal you drew in section 4.1).
- Double click on the new signal's name so that the *Signals Properties* dialog opens, and name the signal **DAC_and_ADC.**



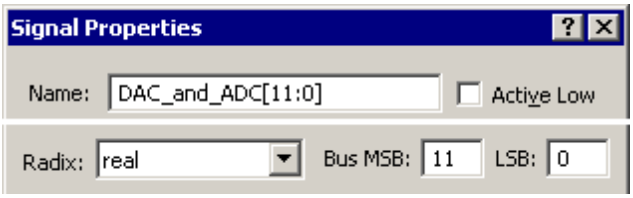
- Next, we will convert this analog signal to a multi-bit digital signal using a 12-bit ADC, by changing the MSB/LSB to **11-0**, and the radix to **Hex**, and then pushing the **Apply** button.

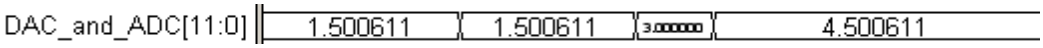




DAC conversion on a multi-bit signal:

- Convert the digital signal, back to an analog signal by changing the radix to **real** and pressing the **Apply** button.





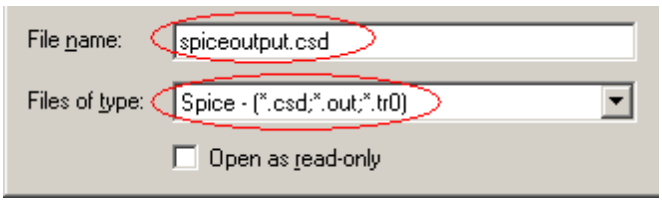
- Notice that the values are no longer exactly 1.5 or 4.5. This is caused by the rounding errors of the 12-bit ADC.

Save the timing diagram:

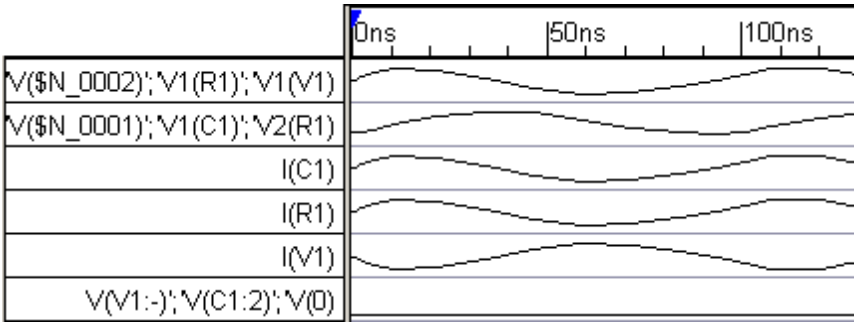
- Use the **File > Save Timing Diagram** to save your tutorial diagram. The next step will be loading a new diagram.

ADC conversion on a single-bit signal:

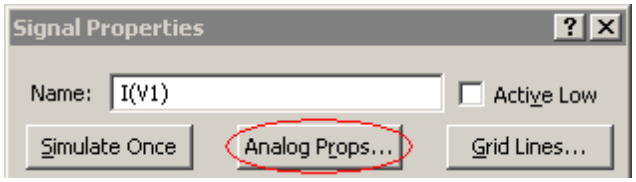
- Choose the **File > Open Timing Diagram** menu to launch the *Open File* dialog. Use the browse button to navigate to the **SynaptiCAD > Examples > SPICE** directory. This can also be accomplished using the **Import/Export > Import Diagram** menu which will allow selective loading of signals.
- Choose a file type of **Spice - (*.csd; *.out; *.tro).**
- Select the **spiceoutput.csd** file, and press the **Open** key load the file.



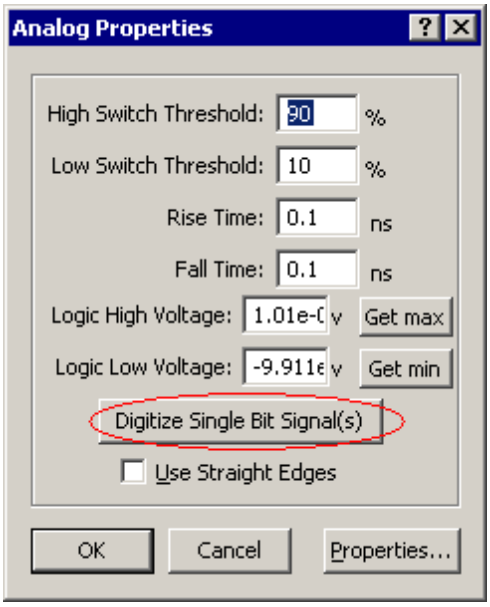
Copyright © 2014, SynaptiCAD Sales, Inc.

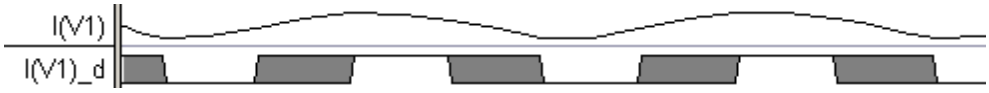


- Double click on **I(V1)** to open the *Signal Properties* dialog. Then press the **Analog Props** button to open that dialog.



- Press the **Digitize Single Bit Signal(s)** button to digitize the signal.
- Notice the waveform display now has two I(V1) signals. The **I(V1)_d** is the digitized version, and **I(V1)** preserves all of the analog data. You can alter the **High Switch Threshold** and **Low Switch Threshold** parameters to change the amount of uncertainty generated on the digital signal's edges. The Logic High and Low Voltage values should be set to the nominal high and low voltage values for the logic type of the digital signal.

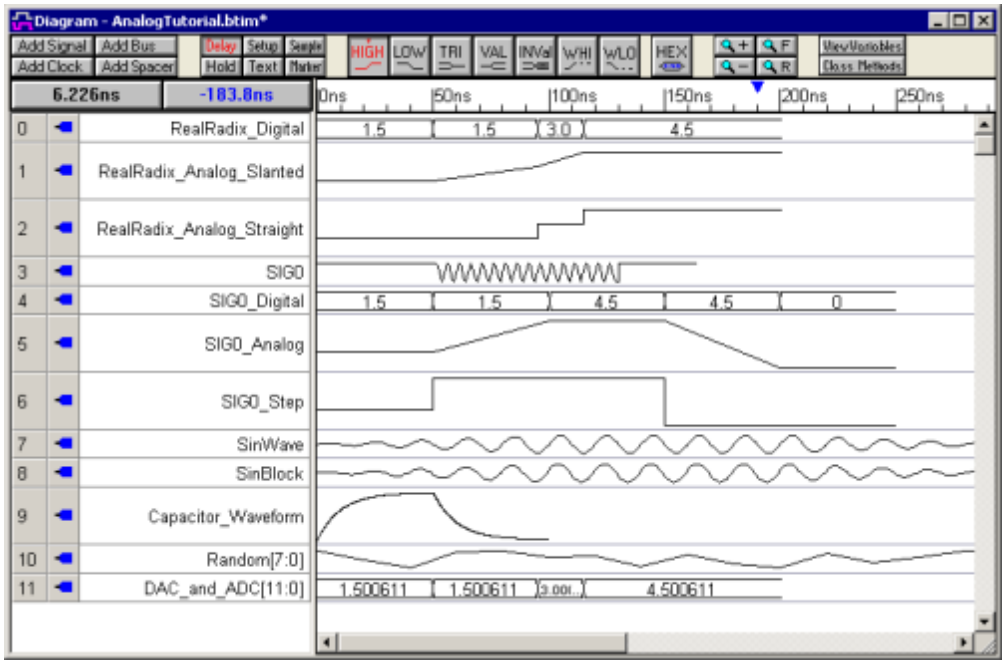




(TD) 4.12 Summary of Analog Signals Tutorial

Congratulations you have completed the Analog Signals Tutorial. You have drawn and generated the following waveforms:

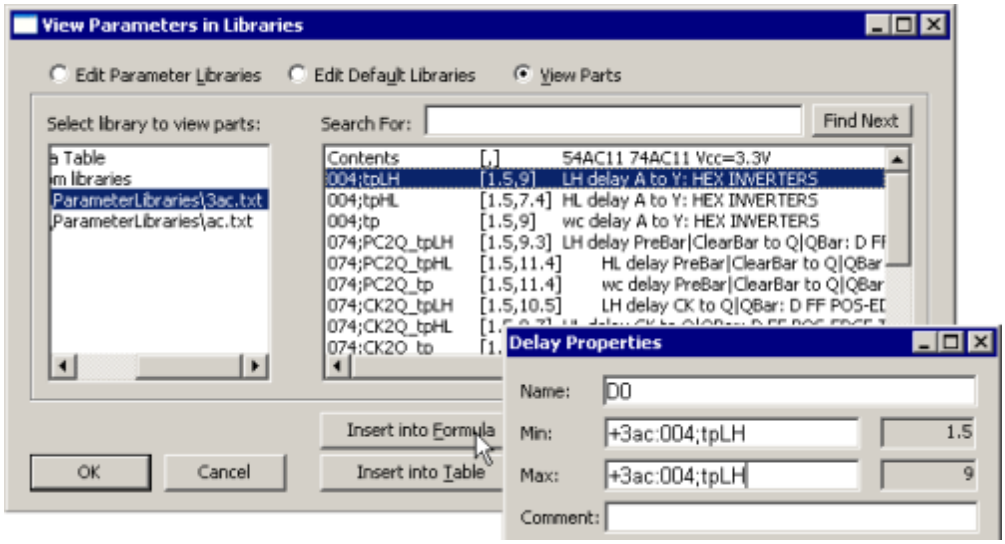
Copyright © 2014, SynaptiCAD Sales, Inc.



Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 5: Parameter Libraries

This tutorial explains how to create and use timing parameter libraries. Library files contain the timing parameter information for circuit components.The timing diagram editor can be used to create libraries with parameters that are exclusive to your projects. The timing diagram editor also ships with several standard libraries that contain over 10,000 timing parameters, and it also supports the industry standard TDML on-line component information.

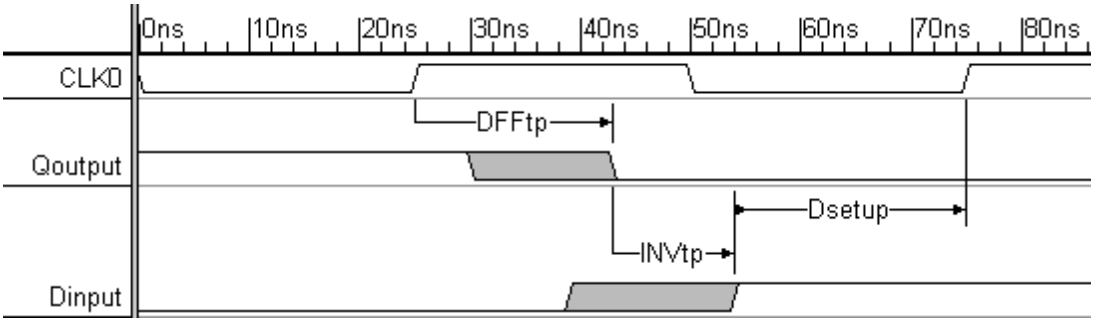


(TD) 5.1 Setup for Library Tutorial

Since this tutorial focuses on how to make and work with libraries, we will save some time by loading a pre-drawn timing diagram.

Load the timing diagram file tutlib.btim:

- Select the **File > Open Timing Diagram** menu option and load **tutlib.btim** from the **SynaptiCAD\Examples\TutorialFiles\ParameterLibraries** directory.
- Select the **File > Save As** menu option and save the file as **mylib.btim** (this will keep the original file intact).



Copyright © 2014, SynaptiCAD Sales, Inc.

Setup the Drawing environment:

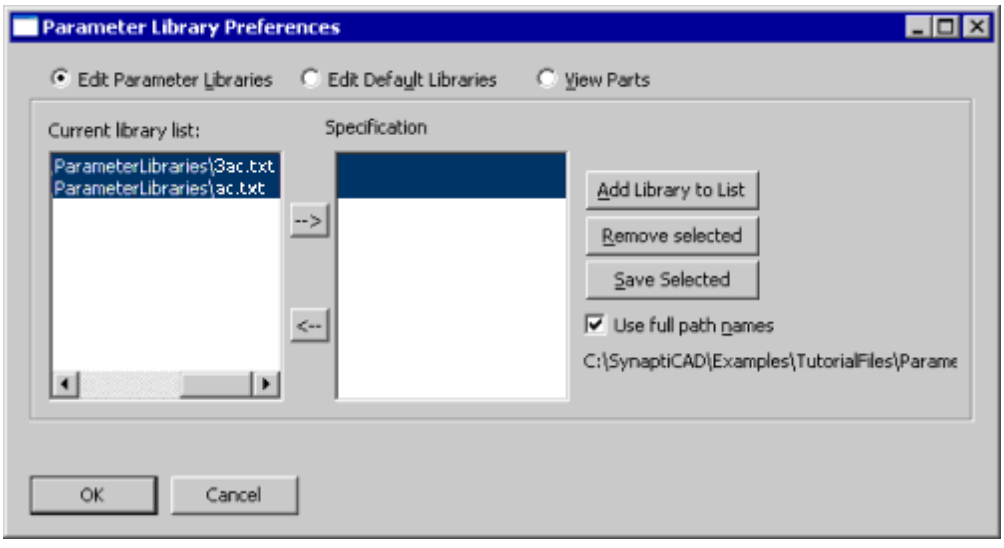
- Minimize the Report window (and Project window if applicable). They are not used in this tutorial.
- Select the **Window > Tile Horizontally** menu option. Both the *Diagram* and the *Parameter* windows should be visible during this tutorial. If you are unable to view one of the windows, use the **Window > Parameter** or **Window > Diagram** menu option to open the missing window.



(TD) 5.2 Add Libraries to the "Library Search List"

In order for a timing diagram to use a library, it must know the library's name and path location. This information is kept in the diagram's library search list.

- Choose the **ParameterLibs > Parameter Library Preferences** menu to open the *Parameter Library Preferences* dialog.
- Press the **Add Library to List** button to open the *Parameter Library Browse* dialog to search for libraries on your disk.
- Select the two sample libraries **ac.txt** and **3ac.txt**, located in the **SynaptiCAD\Examples\TutorialFiles\ParameterLibraries** directory.
- Press the **OK** button to add the selected files to your search list, and close the *Parameter Library Browse* dialog.



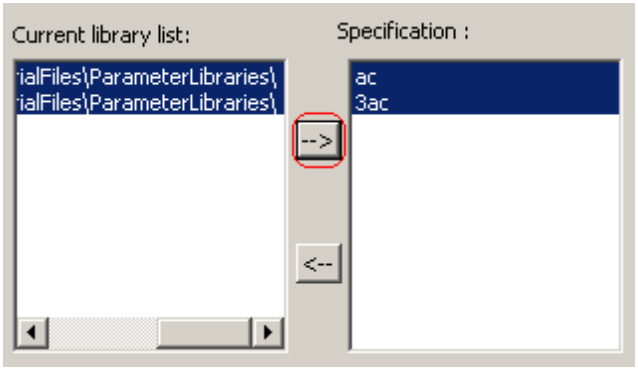
- Notice that the filenames for the libraries have their path names attached unless you have unchecked the **Use full path names** check box. If it is unchecked, the library path searched will be relative to the location of the timing diagram.
- The next section also uses the *Parameter Library Preferences* dialog so leave it open.

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 5.3 Setup the Library Specifications

Often several libraries will contain parameters with the same names, and the diagram needs a way to distinguish between such parameters. SynaptiCAD uses library specifications to make the distinction. In this tutorial the 3ac and ac libraries contain parameters with the same names, so we will use library specifications when we reference these parameter names. If no specification is used when referencing a parameter, the program will use the values of the parameter in the first library that it finds a definition.

- Select both the **3act.txt** and the **ac.txt** library files.
- Press the right arrow button to assign specifications to the selected libraries.
- The specification for ac.txt is **ac** and the specification for 3ac.txt is **3ac**.



Library specification can be remove by selecting them and using the left arrow. You can experiment with this, but make sure the ac and 3ac specifications are assigned before moving on to the next section.

(TD) 5.4 Investigate Preferences Dialog

The *Parameter Library Preferences* dialog has three radio buttons across the top that control what the dialog edits. So far we have been using the dialog with the **Edit Parameter Libraries** radio button selected, which means that we were just editing library settings for the current timing diagram.



Default Settings for new timing diagrams:

If we had selected the **Edit Default Libraries** radio button, then the libraries settings would have been saved in the ini file of the program. All new timing diagrams would automatically start with the default library settings. We will not demonstrate this in the tutorial.

View Parts mode:

Once the libraries are in the library list and have their specifications defined, you will be working with the dialog in the **View Parts** mode. You can select the radio button in the current dialog and continue with the tutorial. However, we want to demonstrate other ways to open the dialog, so close it for now.

- Press the **OK** button to close the *Parameter Library Preferences* dialog.

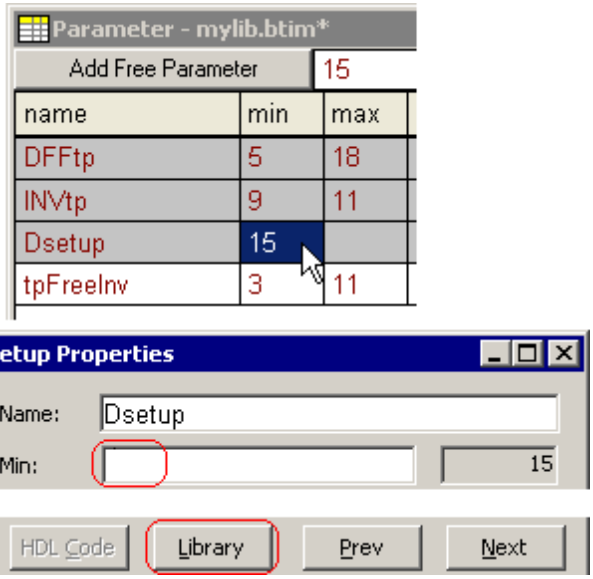
(TD) 5.5 Referencing Parameters in Libraries

Now that we have added the libraries and set the specifications, we want to reference the library parameters in our project.

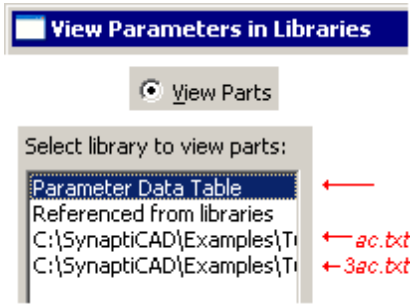
Copyright © 2014, SynaptiCAD Sales, Inc.

Make Dsetup reference a Library part:

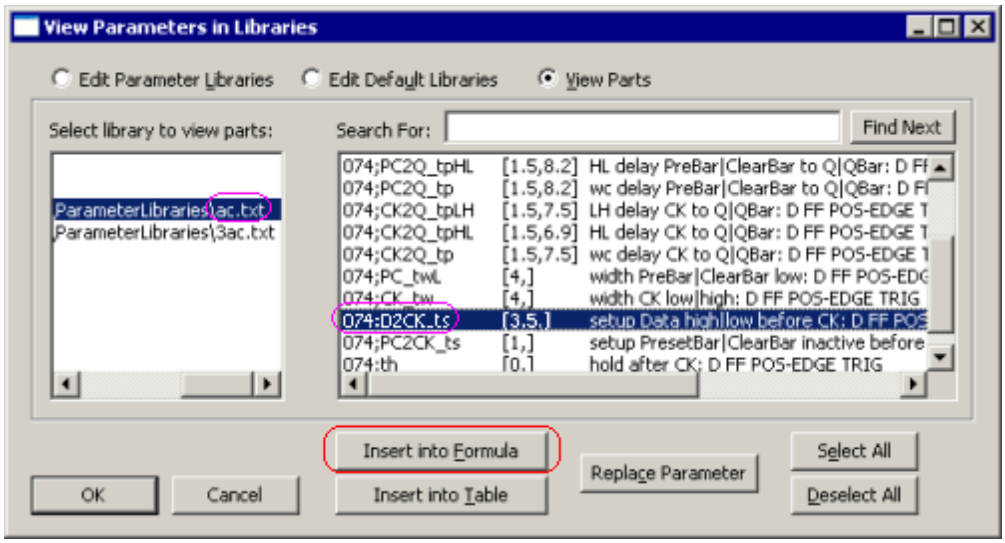
- In the Parameter window, double click on the min value of the Dsetup parameter to open the *Parameter Properties* dialog.
- If there were more than one instance of Dsetup in the diagram, opening the *Parameter Properties* dialog this way would change the timing values for all instances of the parameter.
- Delete the value in the **min** edit box.
- Press the **Library** button to open the *View Parameters in Libraries* dialog (this is really the *Parameter Library Preferences* dialog in **View Parts** mode).



- Notice that there are **three** libraries on the library list; the 3ac.txt and Ac.txt that you added in step 5.2, and one called **Parameter Data Table**. This extra library is a virtual library that lists all the parameters in the current timing diagram. You can use virtual library parameters in formulas just like regular library parameters.



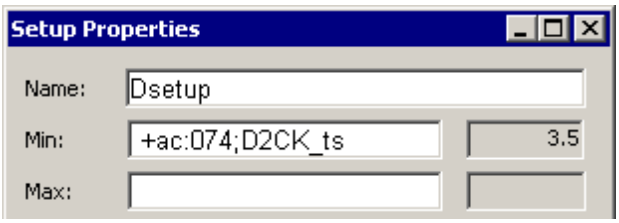
- Select the **Ac.txt** library from the library list. This displays the parameters in this library in the library parts list on the right.



- Scroll down in the library parts and select parameter **074;D2CK_ts**.

Copyright © 2014, SynaptiCAD Sales, Inc.

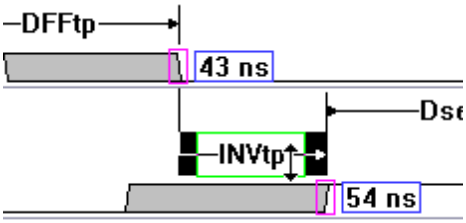
- Press the **Insert Into Formula** button then press the **Ok** button to close the dialog.
- Notice that the min box in the *Parameter Properties* dialog has the parameter with the library specification appended to the front. A "+" sign is also added, so you can easily add together several library timing parameters by selecting multiple parameters then pressing the **Insert Into Formula** button.



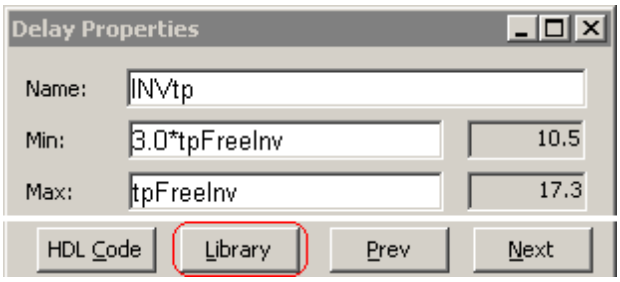
- Notice that the max column is blank, this is because the parameter in the library only has a min value defined.

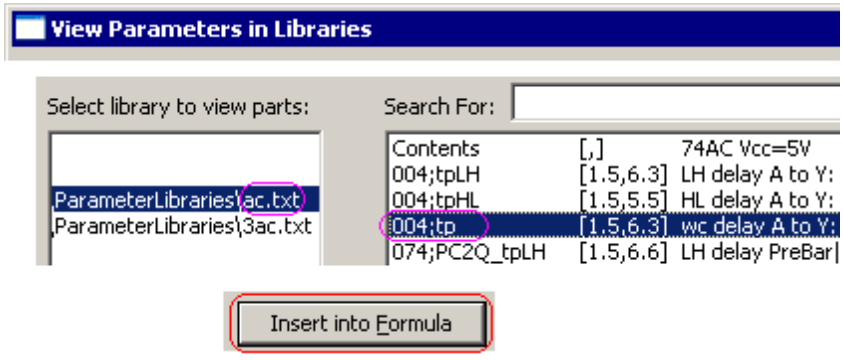
Make INVtp reference a library parameter:

- Double click on the **INVtp** parameter in the diagram window to open the *Parameter Properties* dialog.



- Press the Library button to open the *View Parameters in Libraries* dialog.

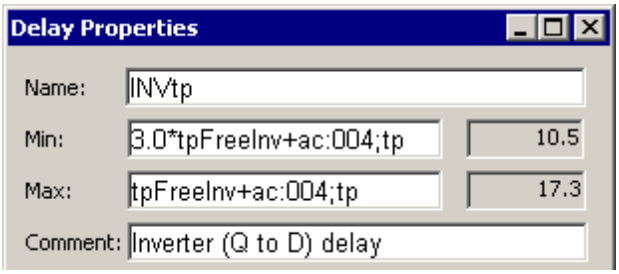




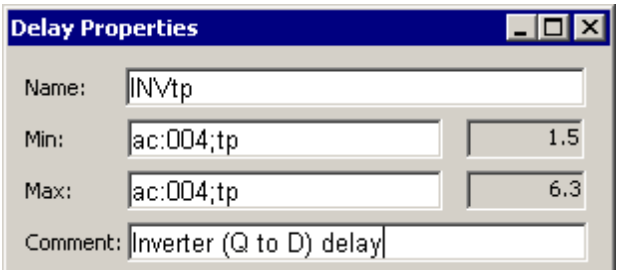
- Select the **ac.txt** library to display its parts on the right hand side.
- Select parameter **004;tp** then press the **Insert Into Formula** button to insert the values into the INVtp boxes.
- Press the **OK** button to close the *View Parameters in Libraries* dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Notice that the **ac:004;tp** parameter was added to the values that were already in the min and max edit boxes.

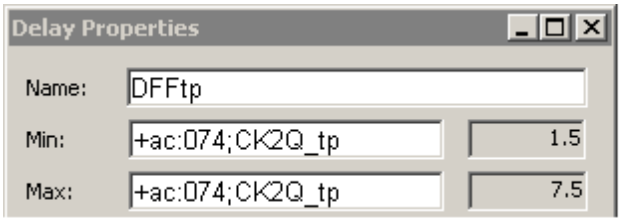


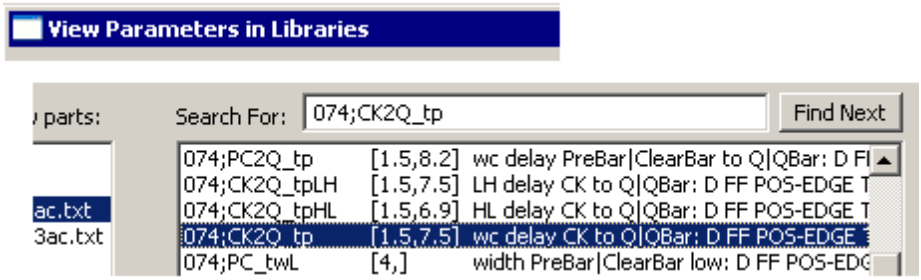
- Delete the original values from the min and max edit boxes, leaving only the **ac:004;tp** value.
- Press the **OK** button to close the *Parameter Properties* dialog.



Make DFFtp reference a library parameter:

- Repeat the above process for DFFtp, by inserting **074;CK2Q_tp** from the **ac.txt** library. Try using the **Search For** edit box in the *View Parameters in Libraries* dialog, instead of scrolling, to find a parameter name.





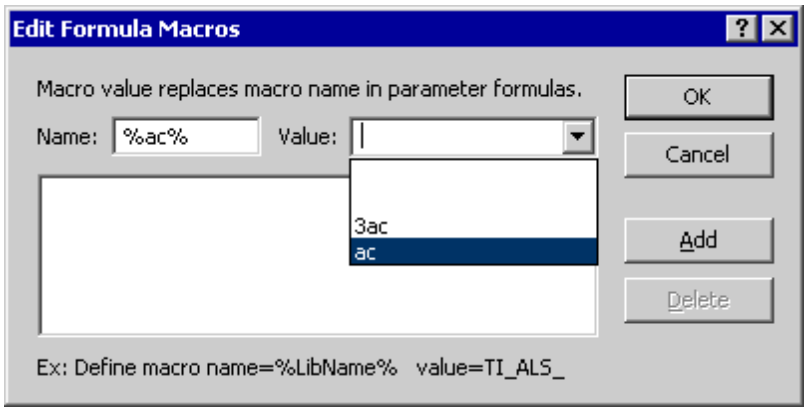
(TD) 5.6 Using Macros to Examine Tradeoffs Between Different Libraries

Your diagram is now using values for the AC logic family operating at 5V. If you want to examine the impact of changing your design to 3.3V, you need to change the library specifications of the parameters to "3ac". It can get tedious changing back and forth between different libraries when you have to change the name of each parameter. To avoid this you can create a macro which you use in place of the library specification in your parameter names. Then to change libraries you just need to change the value of the macro.

Copyright © 2014, SynaptiCAD Sales, Inc.

To create a macro:

- Select the **ParameterLibs > Macro Substitution List** menu option to open the *Edit Formula Macros* dialog.
- Enter **%ac%** into the name edit box.
- Select **ac** from the **Value** drop down box. The drop down box contains all libraries that have specifications.

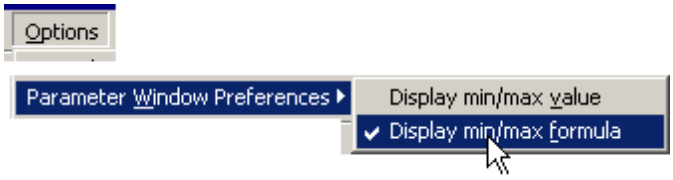


- Press **OK** to add the macro to your macro list.

Edit the Parameters to make them use the macro:

Now we must edit the five min & max values of your parameters, by replacing **ac** with **%ac%**. To make it easier to check our work, first setup the parameter window to display the formulas.

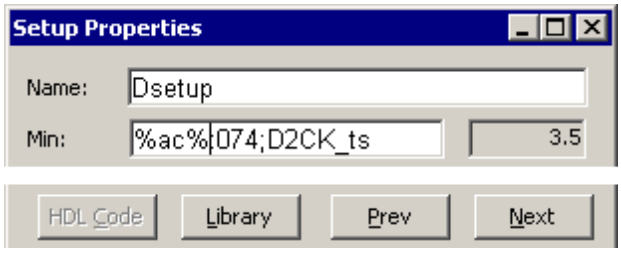
- Choose the **Options > Parameter Window Preferences** menu and check the **Display min/max formula** option.



- Adjust the columns of the parameter window so that you can see the formulas.

name	min	max
DFFtp	+ac:074;CK2Q_tp	+ac:074;CK2Q_tp
INVtp	ac:004;tp	ac:004;tp
Dsetup	+ac:074;D2CK_ts	
tpFreeInv	3	11

- Double click on one of the formulas to open the *Parameter Properties* dialog.
- Replace the **ac** with **%ac%** which references the macro.
- Use the **Prev** and **Next** buttons to move to the next parameter.



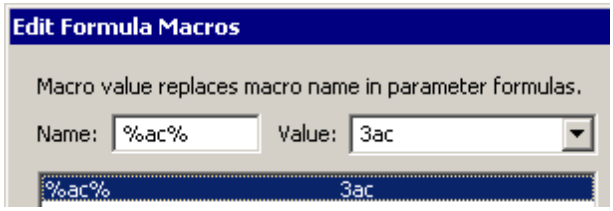
- When you are done the table should look like this:

name	min	max
DFFtp	%ac%:074;CK2Q_tp	%ac%:074;CK2Q_tp
INVtp	%ac%:004;tp	%ac%:004;tp
Dsetup	%ac%:074;D2CK_ts	
tpFreeInv	3	11

Copyright © 2014, SynaptiCAD Sales, Inc.

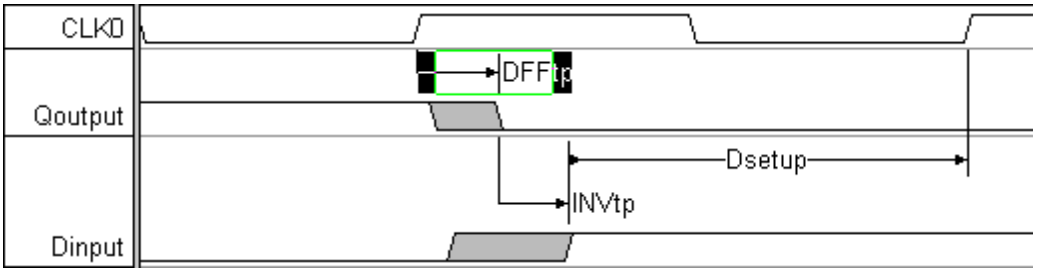
Switch in the 3 volt Library using the Edit Formula Macros dialog:

- Select the **ParameterLibs > Macro Substitution List** menu option to open the *Edit Formula Macros* dialog.
- Click on the macro **%ac%** in the list box. This places this macro into the Name/Value edit boxes.



- Use the **Value** drop down box to change the value of the macro to **3ac**. Click **OK** to close the dialog.

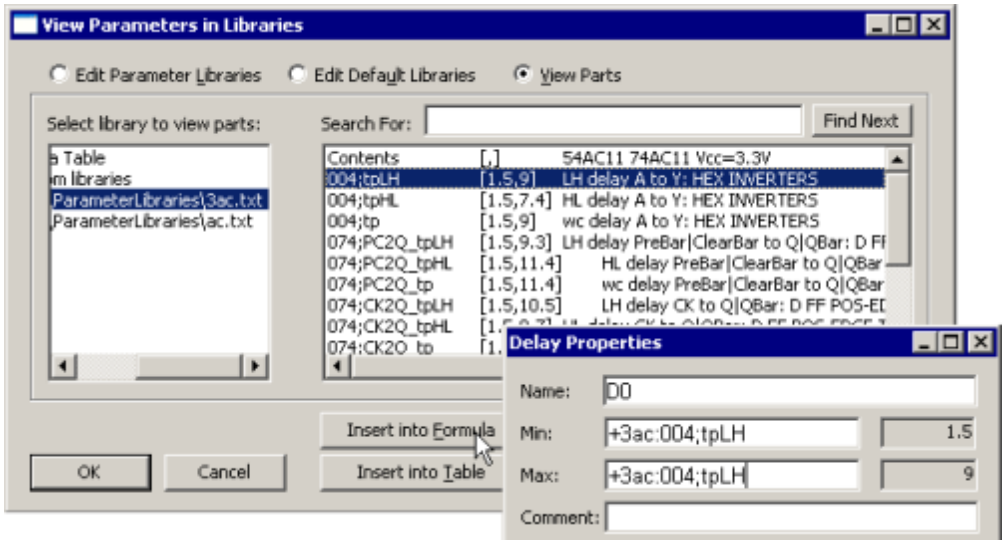
Your design should now be using the 3V AC values (the delays should be longer due to the decreased supply voltage). You have now completed the parameter library tutorial.



Note: Macros can also be used to make short or alternative names for library parameters without having to edit the library names.

(TD) 5.7 Parameter Libraries Summary

Congratulations! You have completed the *Parameter Libraries* tutorial. In this tutorial you experimented with libraries, library specifications, and macros. The Timing Diagram Editing Manual Chapter 10: Parameter Libraries has information on how to create your own libraries.



Copyright © 2014, SynaptiCAD Sales, Inc.

Timing Diagram Editor 6: Advanced Modeling and Simulation

This tutorial demonstrates how WaveFormer Pro can quickly model and simulate a digital system of moderate complexity. We will be modeling a circuit that computes histograms for 64K of data generated by a 12-bit Analog-To-Digital converter (this is a popular method for testing dynamic SNR for ADCs). This circuit is a simplified form of a real VME board that would take several months to model and simulate using conventional EDA tools. Using WaveFormer, we will model and simulate this simplified circuit in 20 minutes. The full circuit with the complete VME bus interface protocol could be modeled and debugged in about 4 hours.

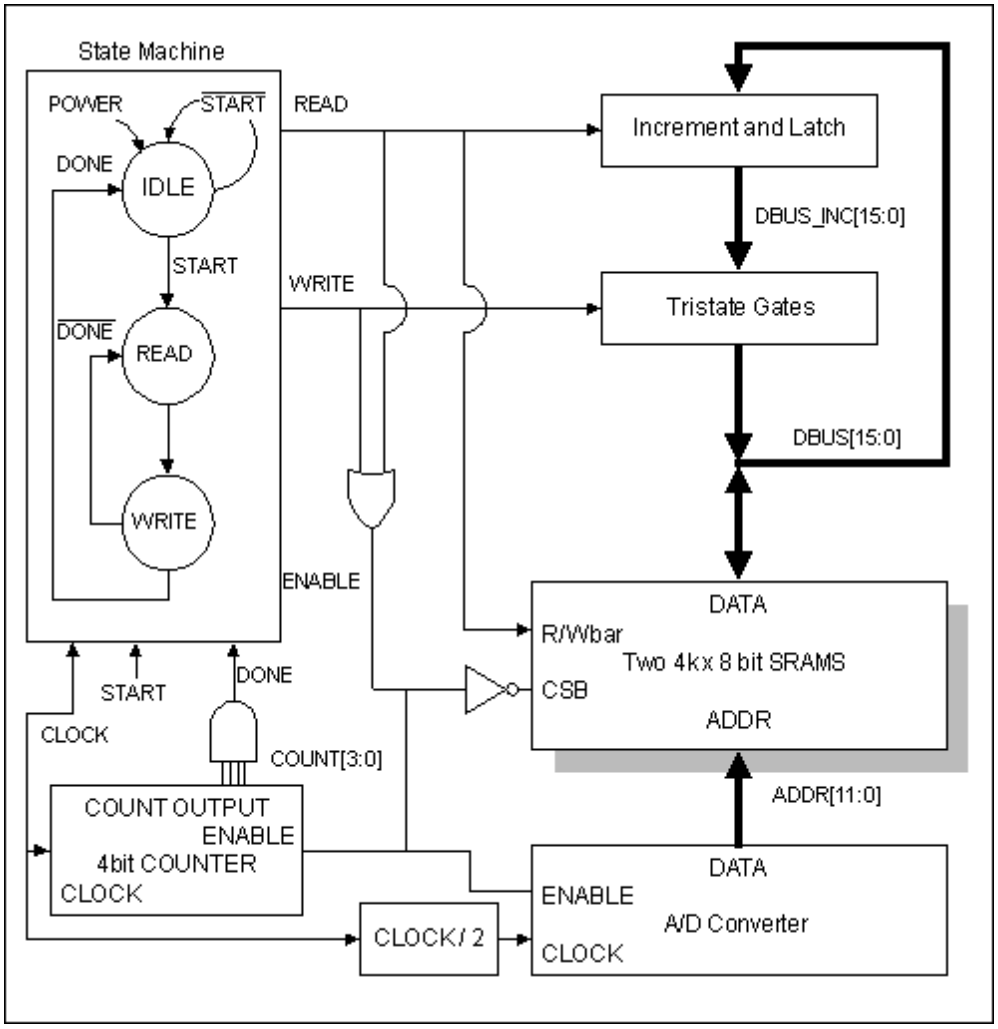


Figure 1: Histogram circuit block diagram.

This tutorial teaches the user how to:

1. Model state machines using the Boolean Equation interface.
2. Generate input signals using temporal and label equations.
3. Use the simulation log to find design entry errors.
4. Simulate incrementally by temporarily modeling outputs as drawn inputs.

Copyright © 2014, SynaptiCAD Sales, Inc.

5. Enter direct HDL code for simulated signals.
6. Use external HDL source code models.
7. Model tri-state gates using the conditional operator.
8. Model n-bit gates using reduction operators.
9. Model transparent latches.
10. Debug Verilog source code using $display statements.
11. Control length of simulation time using a Time Marker.
12. Edit an external HDL file with WaveFormer's Report window.

Before you begin the tutorial you may wish to view Figure 3 in Section 13 which shows a completed version of the diagram that we will generate. File tutsim.btim included in the product directory is a finished tutorial file. You will not use this file during the tutorial itself, but you can always refer back to this file if you encounter any problems during the tutorial.

Circuit Operation

A histogram is a graph displaying the count of same 12-bit values received from the ADC. To store the histogram count values we will use a 4K SRAM (2 12 storage cells) to hold a count for each possible 12-bit value that the ADC can generate. The width of the SRAM depends on how many data values we will accumulate from the ADC. In the worst case, the ADC could generate the same value for the entire histogram accumulation, so the SRAM must be able to store a value of up to 4K. Thus we will use 2 8-bit wide SRAMs (2 16 = 64K > 4K).

When the circuit starts operation, the SRAM should contain zeros at every address. Each time a data value is generated by the ADC, that data value is used as an address to look up the current count for the data value in the SRAM. The count is incremented by one and the new value is written back to the SRAM. This continues until the circuit has r

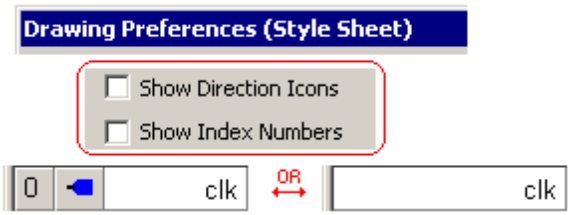
(TD) 6.1 Set up a New Timing Diagram

Create a new timing diagram:

- Select the **File > New Timing Diagram** menu option to create a new diagram.
- Minimize the *Parameter* window. It is not used in this tutorial.
- Select the **Window > Tile Horizontally** menu option. This will provide us with optimal viewing by rearranging the *Diagram* window and the *Report* window (if either of these windows is not visible, select the menu option **Window > Diagram** or **Window > Report** to make it visible).

Hide the Direction and Index Columns in the Label window:

- Choose **Options > Drawing Preferences** to open the dialog. Then uncheck **Show Direction Icons** and **Show Index**.



Now that we have a new diagram to work with, we are ready to model the components of our circuit.

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 6.2 Generate the Clock, Draw Waveforms, & Use Waveform Equations

The histogram circuit has a system clock, CLK0, and three signal inputs, POWER, START and ADDR. We will create the waveforms for each of these signals using three different methods: generating from clock parameters, drawing waveforms by hand, and automatically generating waveforms from temporal equations.

2.1 Automatically generate the CLK0 system clock

Add a clock named CLK0 with a period of 100 ns:

1. Click the **Add Clock** button 

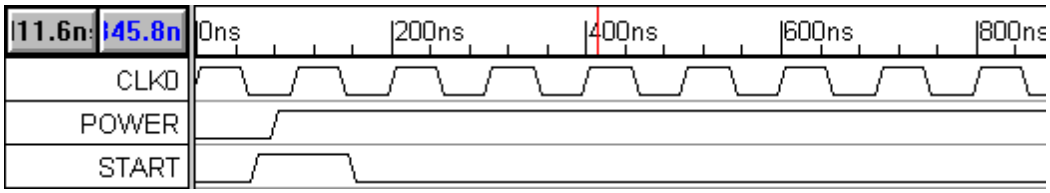
to open the *Edit Clock Parameters* dialog.
2. Verify that the default values are: **name = CLK0**, **period = 100 ns**, and **duty = 50%**. If not then make the necessary adjustments.
3. Press **OK** to accept the default values for the clock.

2.2 Graphically draw the POWER and START signal

The POWER signal is a power-on reset signal that we will use to set the initial state of our state machine. The START signal is an external input to the system that pulses high to initiate acquisition in the histogram circuit. The POWER and START waveforms are relatively simple, so we will draw them with the mouse.

1. Click on the **Add Signal** button 

twice to add two signals.
2. Double-click on a signal name to open the *Signal Properties* dialog. Use this dialog to change the names of the signals to **POWER** and **START**.
3. Draw the *POWER* signal so that it is low for 80ns, then high for 2000ns.
4. Draw the *START* signal so that it is low for 60ns, high for 100ns, and then low for 800ns:
5. Verify that the timing diagram looks like:



Waveform drawing and editing techniques can be found in *Chapter 1: Signals and Waveforms* in the online help.

2.3 Use Temporal and Label Equations to model ADDR (A/D converter's output data)

We will model the A/D converter just as a data source, so all we need to do is generate a *virtual* bus signal called ADDR (the output from the ADC) that drives the address lines of the SRAMs. The ADDR waveform has a regular pattern that can be described easily using an equation, but would be tedious to draw by hand.

Add a virtual bus signal called ADDR:

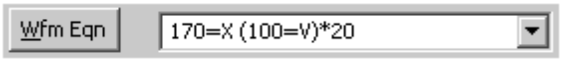
1. Add a signal and change the name to **ADDR**. Leave the *Signal Properties* dialog open for the rest of the section.
2. Set the signal's **Radix** to **hex** and the **MSB** to **11**. Changing the MSB and Radix defines *ADDR* as a 12-bit signal that display its values in hexadecimal format.

Copyright © 2014, SynaptiCAD Sales, Inc.



The A/D converter is driven by a clock that is 1/2 the frequency of the state machine clock *CLK0*, so the *ADDR* value should change every other clock cycle (this maintains the same address for the read out of each RAM cell's count data and its write back after it is incremented). The *ADDR* signal should be **unknown for 170ns** then it should have **twenty valid states, each 200ns** in duration. Use the **Waveform Equation** interface of the *Signal Properties* dialog to generate the ADDR waveform:

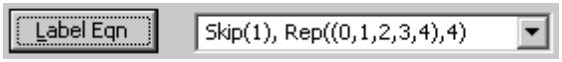
1. Enter the following equation into the edit box next to the **Wfm Eqn** button: **170=X (200=V)*20**



2. Press the **Wfm Eqn** button to apply the waveform equation. Notice that the waveform drew itself. If the waveform didn't draw, a syntax error was made when typing in the equation. To determine what the error was, look at the file waveperl.log displayed in the Report window. This file will show you which part of the equation could not be parsed. Fix the error, and press the **Wfm Eqn** button again.

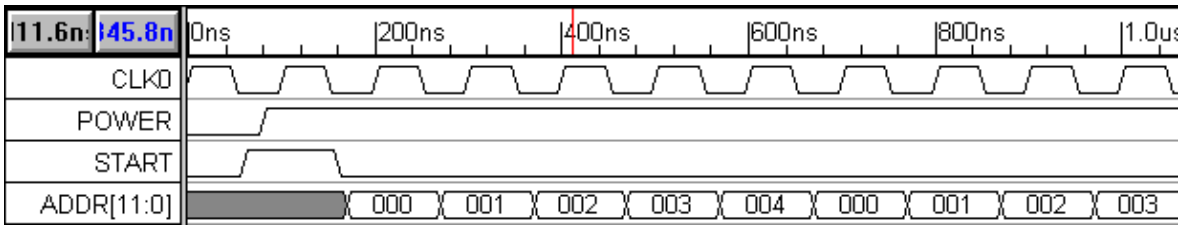
Next, we will label the states of the *ADDR* bus using a **Label Equation**. Each state could be labeled individually using the extended state field of the HEX dialog box, but labeling twenty states would take a long time. Instead, we will write an equation to label all the states at once. *Chapter 11* covers all the different state labeling functions.

1. Enter the following equation into the edit box next to the **Label Eqn** button **Skip(1), Rep((0,1,2,3,4), 4)**.



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

This equation will generate a hex count from 0 to 4, and then repeat it 4 times. The Skip(1) means start labeling after the first state (which we defined to be an invalid state using our waveform equation). Your timing diagram (at the appropriate zoom level) should now resemble the diagram below.



(TD) 6.3 Modeling State Machines

We will use a simple one-hot state machine to control the circuit, and we will model it using Boolean Equations. A one-hot state machine uses a single flip-flop for each state. At any given time, only the flip-flop representing the current state will contain a 1, the other flip-flops will be at 0 (hence the name one-hot).

Copyright © 2014, SynaptiCAD Sales, Inc.

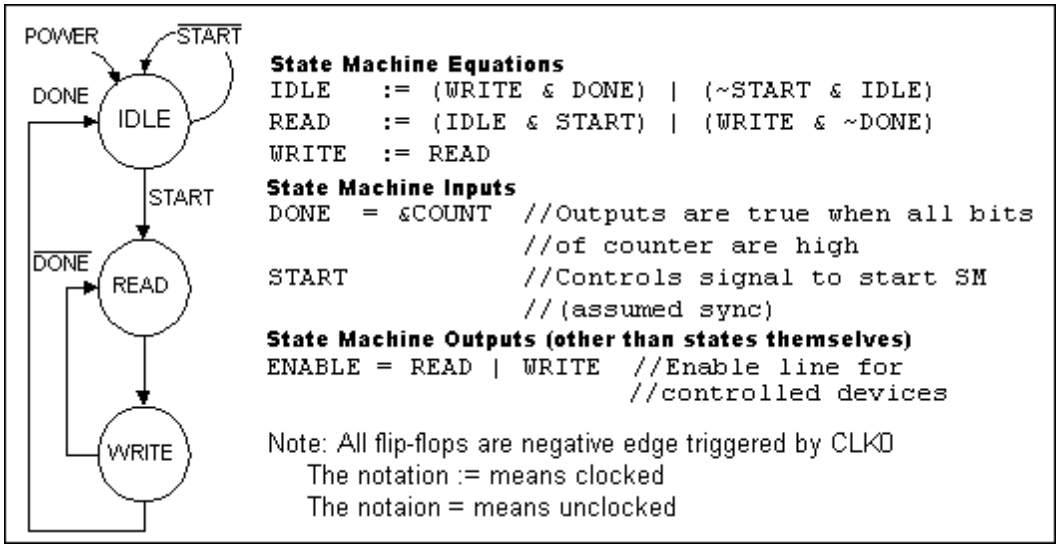
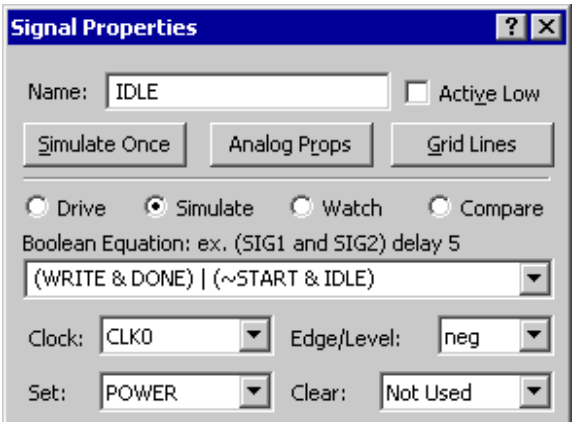


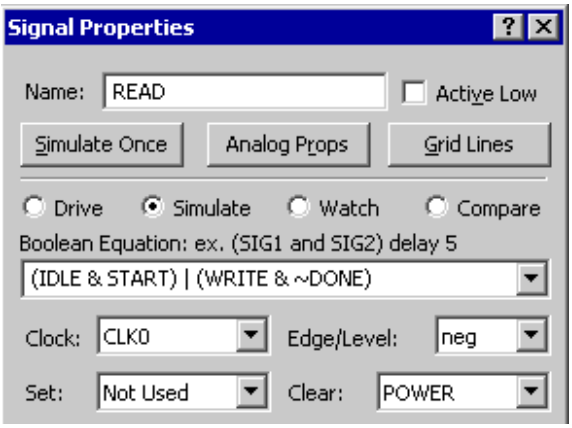
Figure 2: State diagram and design equations for the histogram controller state machine

The state machine (SM) initializes to the **IDLE** state. On the negative edge of the clock after **START** goes high, the SM will enter the **READ** state and look up the current count for the current address value being output by the A/D converter. This value will be incremented by a simple fast-increment circuit. On the next clock, the SM will enter the **WRITE** state, latching the incremented value into a transparent latch called DBUS_INC and initiating the write back of the incremented data to the SRAM. The state machine will continue to toggle between the **READ** and **WRITE** state until the desired number of data values have been histogrammed (determined by the size of the binary counter called COUNT), at which point the SM will return to the **IDLE** state. Figure 2 shows the SM that we will model.

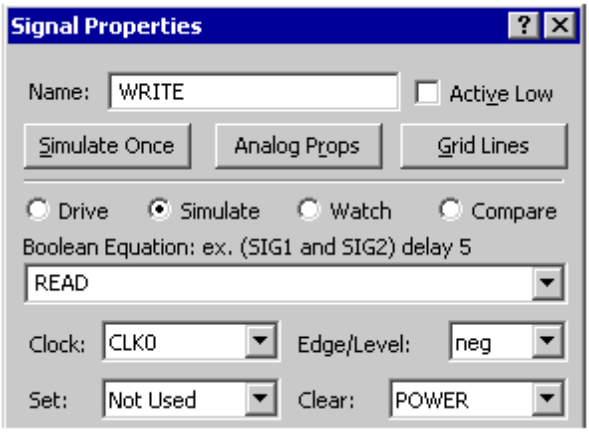
The state machine is modeled in WaveFormer using one signal for each state. Next we will enter the equations for the state machine, however these signals are **not simulated until Section 5** because signal **DONE** has not yet been defined.

1. Add 3 signals and name them **IDLE**, **READ** and **WRITE**.
2. For each signal, enter state machine Equation, select Simulate button, setup the clock and trigger edge, and setup the set and clear signals as shown in the following pictures:





Copyright © 2014, SynaptiCAD Sales, Inc.



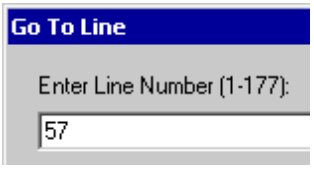
Notice the display Compile Error in the bottom right hand corner and notice that the state machine signal names turned gray . This is because the IDLE and READ equations reference a signal called DONE. This signal has not been defined so if you try to simulate you get errors. In the next section we will investigate the different ways to detect and fix simulation errors.

(TD) 6.4 Checking for Simulation Errors

If you check the simulator log file, **simulation.log** simulation.log in the *Report* window, you will see an error message reporting that DONE is not declared. The log file also reports the lines in the WaveFormer-generated Verilog source code file where this error occurred. The WaveFormer-generated source file will have the same filename as your diagram, but with a file extension of **.v** instead of **.btim** (so if your diagram is untitled.btim, the source code file is untitled.v). This source file is automatically opened by the *Report* window whenever WaveFormer Pro generates this file (by default this occurs every time you make a change to your design while simulating signals).

View the HDL lines where the errors occur:

1. Check the log file for the line number at which the error(s) occurred. In the *Report* window, click on the **simulation.log** tab . When we ran the simulator, our error occured at line number 57 (your run may be different) , as indicated by the error message: **C:\SynaptiCAD\UNTITLED.v: L57: error: 'DONE' not declared**
2. Click on the tab for the ***.v** file at the bottom of the Report window. This will open your source file in the *Report* window.
3. Click inside the *Report* window, and press **<Ctrl>-G**. This brings up the *Go To Line* window. Enter 57 as the line number you wish to jump to, and press **OK**.
4. As expected, these lines show the HDL code that simulates the IDLE and READ signals.



NOTE: Do not make changes in this source file as your changes will automatically be overwritten the next time a simulation is performed; instead, we will make the appropriate changes in the *Diagram* window and *Signal Properties* dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.

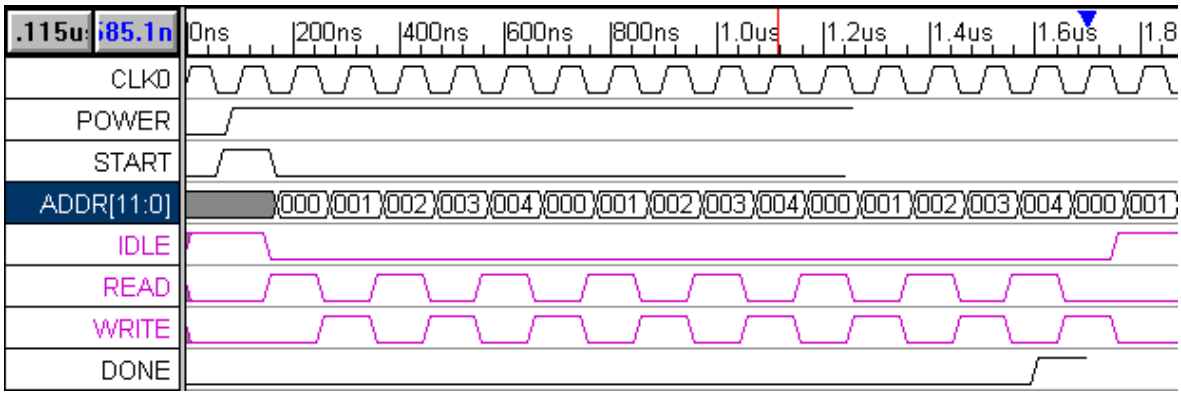
(TD) 6.5 Incremental Simulation

One common problem in simulating and debugging digital systems is that large parts of the design have to be entered before testing can begin because the parts provide input to each other. One solution is to break a design up into pieces and test each piece with test vectors that represent the output of the other pieces. However, generation of the test vectors can be time consuming.

SynaptiCAD products provide a very simple and quick method for testing small parts of a design: graphically draw the signals for the missing parts of the design to test the design at its current state of development. Then later add the design information that models these signals (in other words, we temporarily model simulated outputs as drawn inputs).

We will now use this method below to verify the operation of our state machine before we enter the HDL code that generates the DONE signal:

1. Add a signal called **DONE**.
2. **Draw** a **low** segment for 1.6 us, followed by high pulse that lasts for at least one clock cycle. Click on **Apply** to run the simulation.
3. The diagram should now show the simulation output from your state machine. The simulated signals are pink to distinguish them from graphically drawn signals.



Make sure everything is working properly:

1. First make sure that the simulation status indicators read Simulation Good Simulation Good. If the indicators still show an error, then the **simulation.log** file will help you to pinpoint the error in your diagram.
2. Next, check your diagram against the figure above to verify that your state machine is simulating correctly.
3. If the simulation succeeded and there are still discrepancies in the output, check your design equations and the input stimulus you've drawn (START and DONE signals).

Once you have the circuit simulating properly, let's see what happens if the *START* pulse gets too small:

1. **Drag** the falling edge of the *START* pulse back to approximate **140 ns** (before the falling clock edge at 150 ns). This step causes the state machine to stay in the IDLE state (the IDLE signal stays high).
2. **Double click** on the falling *START* edge and enter a time of **160** into the *Edge Properties* dialog to restore proper operation.

Copyright © 2014, SynaptiCAD Sales, Inc.

(TD) 6.6 Modeling Combinational Logic

In addition to the state signals, the state machine has one other output signal called ENABLE that is used to enable the SRAM, the DONE counter, and the ADC. ENABLE is just the output of an OR gate with the READ and WRITE signals as inputs. In Section 3 we used the Boolean Equation interface to model the flip-flops of the state machine. We will use the same interface to model combinatorial logic. To do this choose the default clock called unclocked. If a signal other then unclocked is selected, then the Boolean Equation interface models registers or latches depending on the type of Edge/Level trigger selected. Chapter 12 covers the advanced features of the Boolean Equation interface including the min/max delay features.

Model the Enable logic:

1. Create a new signal called **ENABLE**.
2. Enter the equation: **`READ | WRITE`** into the **Boolean Equation** edit box in the *Signal Properties* dialog.
3. Check the **Simulate** radio button.
4. Verify that *ENABLE* is the **OR** of *READ* and *WRITE*. If ENABLE did not simulate, use the techniques found in section 4 to find your error. Remember that signal names are case-sensitive.
5. Click **OK** to close the dialog.



(TD) 6.7 Entering Direct HDL Code for Simulated Signals

For simplicity, the counter output COUNT is modeled using a simple block of behavioral HDL Code instead of using Boolean equations. It would take a large number of Boolean equations to model the counter and the equations would be difficult to modify if the counter operation had to be changed. For this tutorial we will create a 4-bit counter to test our system. This counter could be easily modified later to make it 12-bit (to acquire 4K worth of data). To enter direct HDL code for the COUNT signal:

1. Create a signal called **COUNT**.
2. In the *Signal Properties* dialog, set the Radix to **hex**, its MSB to **3**, and check the **Simulate** radio button.
3. Press the **Verilog** radio button 

to switch from the Equation view to the HDL Code view/ editor.
4. **Enter** the Verilog code below in the HDL Code editor of the *Signal Properties* dialog (comments begin with // and can be skipped during code entry). You can copy and paste the text into WaveFormer instead of typing it (Select and copy to clipboard the source code below, then click into the HDL Code window in WaveFormer and press **<Ctrl>-V** to paste the text):

```
reg [3:0] COUNTER;    //declare a 4-bit register called COUNTER
always @(negedge CLK0)        //on each falling edge of CLK0
  begin
```

Copyright © 2014, SynaptiCAD Sales, Inc.

```
    if (ENABLE)
      COUNTER = COUNTER + 1;  // count while ENABLE is high
    else
      COUNTER = 0;    // synchronous reset if ENABLE is low
    end
assign COUNT = COUNTER;   //drive wire COUNT with reg COUNTER value
```

5. Click the **Simulate Once** button to simulate the **COUNT** signal.

Note: All signals in WaveFormer are modeled as wires, so the assign is required at the end of the HDL code block to drive the COUNT wire with the value of COUNTER (which must be a register in order to remember its value).

To increase the size of the counter to acquire 4K data values (**do not do this now**), we could change the MSB of COUNT to 11 and change the declaration of COUNTER in the HDL code to:

```
reg [11:0] COUNTER;   //example only, don't do in this tutorial
```

[****] 106

(TD) 6.8 Modeling n-bit Gates

Next we will model the **DONE** signal that we originally drew as an input to the state machine. The DONE signal is generated by performing a bitwise AND of the **COUNT** signal (we are done whenever all the counter bits are high).

To model the DONE Signal:

1. Double click on the **DONE** signal name to open the *Signal Properties* dialog box.
2. Enter the following equation in the Boolean Equation edit box: **`&COUNT`**
3. Check the **Simulate** radio button. The resulting signal should look like the hand drawn signal except that it is a purple simulated signal.

The **&** operator when used as a unary operator is called a *reduction-AND* operation. A *reduction-AND* indicates that all the bits of the input signal should be ANDed together to generate a single bit output. This is equivalent to the following equation**: `COUNT[0] & COUNT[1] & COUNT[2] & ...`**

One nice benefit of using a reduction operator instead of the above equation is that it automatically scales the circuit to match the current size of the **COUNT** signal (it's also a lot easier to type)!

(TD) 6.9 Incorporating Pre-written HDL Models into Waveformer Simuations

We will use an SRAM HDL module contained in an external file (**sram.v**) to model the SRAM. This model is fairly complex and accurately models the asynchronous interface that is commonly used by most off-the-shelf SRAMs. One special feature is that the SRAM resets all its memory cells to zero when it first starts up. In a real circuit, we would need to add extra logic to iterate through the addresses, writing zeros at each one. A full description of the Verilog modeling of this SRAM is outside the scope of this tutorial, but let's take a quick look at it inside the *Report* window:

1. Select the **Report > Open Report Tab** menu option and open the file **sram.v** (located in the **SynaptiCAD\lib\Verilog** directory). Verify that you can view the file in the *Report* window. Keep this file open because we will be referring back to this file later in the tutorial.

Copyright © 2014, SynaptiCAD Sales, Inc.

9.1 Including an external SRAM Verilog model file into WaveFormer

To add the SRAM model to our design we need to modify the **wavelib_exact.v** file that contains the models used by WaveFormer. The SRAM model code cannot be entered into a signal's HDL code window because the model declares a module and modules cannot be nested in Verilog (WaveFormer puts all the HDL code from signals into a single module called testbed). All user-written Verilog modules should be declared in wavelib_exact.v (or preferably, included from separate files into wavelib_exact.v using the include directive as will be doing). In this case, the source code for the SRAM is already contained in a separate file called sram.v and we only need to add an include statement to wavelib_exact.v to let WaveFormer know about it. To modify the wavelib_exact.v file:

1. Select the menu option **Report > Open Report Tab** and open the **wavelib_exact.v** file in the **SynaptiCAD\hdl** directory.
2. Add the following line to the beginning of the wavelib_exact.v (it may already be there depending on which SynaptiCAD product you are using): **`include "lib\verilog\sram.v"**
3. Select the **Report > Save Report Tab** menu option to save your change.

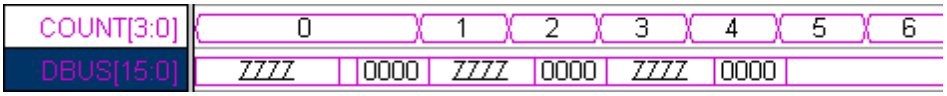
9.2 Instantiating the SRAM component models

To drive the data bus **DBUS**, we need to instantiate two instances of the SRAM model:

1. Create a new signal called **DBUS**.
2. Set the Radix to **hex**, set the MSB to **15**, check the **Simulate** radio button, and select the **Verilog** radio button.
3. Enter the following HDL code into DBUS's HDL code window:

```
wire CSB = !ENABLE;
sram BinMem1(CSB,READ,ADDR,DBUS[7:0]);
sram BinMem2(CSB,READ,ADDR,DBUS[15:8]);
```

The first line creates an internal signal that is an inverted version of the ENABLE line (the SRAM is active low enabled). The next two lines instantiate two 4Kx8 SRAMs and connect up their inputs and outputs (the first SRAM contains the low byte of the count and the second contains the high byte).



(TD) 6.10 Modeling the Incrementor and Latch Circuit

In Section 3 we used the Boolean Equation interface to model the state machine using negative edge-triggered registers. Now we will use the same interface to generate level-triggered latches used to model the increment-and-latch circuit. The value stored in the SRAMs is placed on DBUS and the incrementor circuit takes that value, adds one to it, and latches the incremented value:

1. Create a new signal called **DBUS**_INC.
2. Enter the following equation into the **Boolean Equation** edit box: **DBUS + 1**
3. Choose the **READ** signal from the **clock** drop-down list box.
4. Choose **high** from the **Edge/Level** drop-down list box. This selects the type of latch to be used.
5. Set Radix to **hex**, MSB to **15**, and check the **Simulate** radio button.
6. Press the **Simulate Once** button and verify that DBUS_INC is an incremented version of DBUS. If DBUS_INC did not simulate, use the methods in section 4 to determine the error.

Copyright © 2014, SynaptiCAD Sales, Inc.



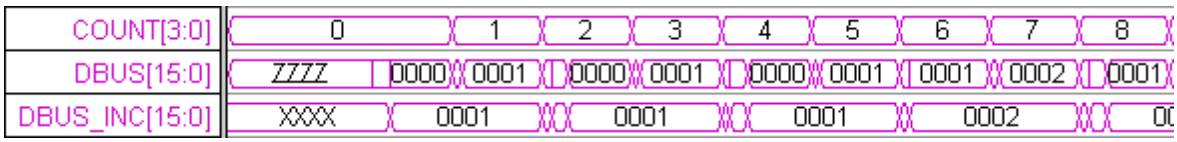
(TD) 6.11 Modeling Tri-State Gates

There are 2 possible drivers for DBUS: the SRAMS which we modeled in section 9, and the tri-stated output of the DBUS_INC signal. All the drivers for a bus should be included in the code for the bus.

To add the tri-state gate to DBUS:

1. Double click on the **DBUS** signal name to open the *Signal Properties* dialog box.
2. In the direct HDL code edit box add a 4th line of HDL code to DBUS:

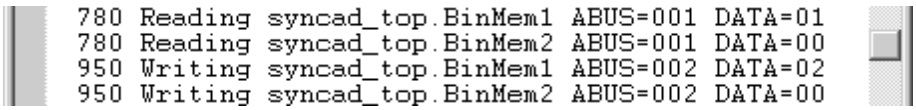
```
assign DBUS = WRITE ? DBUS_INC : 'hz;
```



Line 4 models the tri-state gates that follow the latches in the histogram circuit. These tri-state gates are enabled whenever the WRITE signal is high. We use the conditional operator (condition ? x : y) which acts like an if-then-else statement (if condition then x else y). If WRITE is high, DBUS is driven by DBUS_INC (the incremented version of DBUS that we latched), else the tri-state drivers are disabled ('hz means all bits are tri-stated).

(TD) 6.12 Debugging External Verilog Models

Verilog contains two system tasks (commands), **$display** and **$monitor**, that can be included in Verilog source files for debugging purposes. **$display** acts like a C-language **printf** statement which prints to the simulation log file **simulation.log** whenever it is executed by the Verilog simulator. **$monitor** is similar, but it automatically prints to the log file whenever any of the signals listed in this command change state. The SRAM model file **sram.v** contains two **$display** statements that output the address and data values for the SRAM whenever the SRAM is read from or written to (you can view the **$display** commands in **sram.v** in the *Report* window). You can see the output of the **$display** commands by viewing **simulation.log** in the *Report* window. Each time the SRAM performs a read or write a message is sent to the log file.



(TD) 6.13 Verify the Histogram Circuit

At this point we have modeled the entire histogram circuit, so your diagram should resemble the figure below. If it doesn't, check the **simulation.log** for errors and correct as necessary. The output of the **$display** commands will be particularly useful if you are getting x's on your **DBUS** signal which indicates unknown data is being read from your RAMs. One thing to check for is that your diagram is never performing a write to an unknown address (an address containing x's) in your RAM bank. If you write a value to an unknown address, the memory model has no way of knowing which memory location has been changed. Therefore, all the memory locations in the entire address space of the

Copyright © 2014, SynaptiCAD Sales, Inc.

RAM bank may or may not have been changed. The memory model is forced to represent this unknown state by setting all memory locations in the SRAM to x!

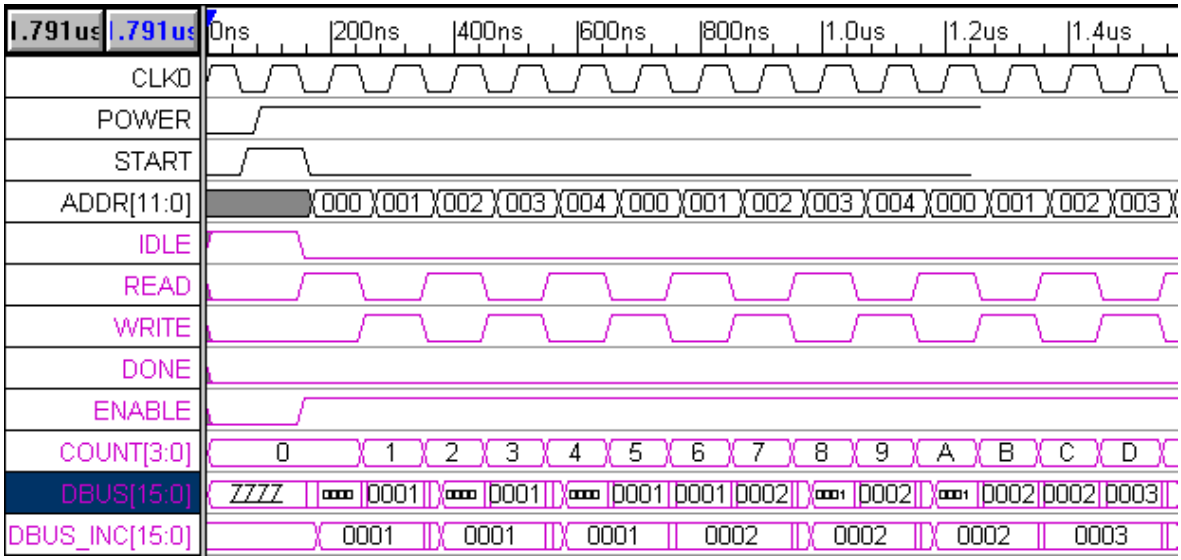
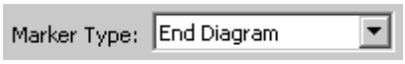


Figure 3: Completed Timing Diagram

(TD) 6.14 Controlling the Length of the Simulation

By default, WaveFormer simulates to the last drawn signal edge. You can also use a time marker to control the length of the simulation. To place a time marker:

1. Click the **Marker** button  found on the button bar. This turns the Marker button red which indicates that right clicks in the *Diagram* window will add marker lines.
2. Right click at about **1us** in the *Diagram* window. A new time marker line will appear.
3. Double click on the marker to open the *Edit Time Marker* dialog.
4. Set the marker type to **End Diagram**. 
5. Click **OK** to close the dialog, then **drag** the marker on the screen. As you move the marker, the simulator will automatically resimulate the design up to the time location of the marker.

(TD) 6.15 Editing Verilog Source Files

To demonstrate how to make changes to a Verilog source file inside WaveFormer, we will edit the SRAM model file sram.v in the Report window:

1. Change **line 18** from: `ram[i] = 0;` To **`ram[i] = 8;`**

   This causes the SRAM cells to be initialized with 8 instead of zero.
2. Select the **Report > Save Report Tab** menu option to save your change.

   Let's see the effect of this change:
3. Press the **Simulate Once** button in the *Signal Properties* dialog, or move an input edge. Either of these steps initiates a resimulation.

You may have anticipated that DBUS would now show 8 (we did when we first did this tutorial!), but it

Copyright © 2014, SynaptiCAD Sales, Inc.

is correct in showing 808 because our DBUS is a 16-bit value composed of the data in two parallel SRAMs each initialized with 08 (hence 0808 = 808).

4. Reset the line back to **ram[i] = 0;**

(TD) 6.16 Simulating Your Model with Traditional Verilog Simulators

The Verilog model of your system created by WaveFormer can also be simulated by traditional Verilog simulators. The complete verilog model simulated by WaveFormer is composed of (1) the verilog file generated by WaveFormer (**untitled.v** for this tutorial), (2) the WaveFormer library file **wavelib.v**, and (3) any external model files you have included (e.g. **sram.v** for this tutorial). Follow the instructions of your Verilog simulator to simulate these files together.

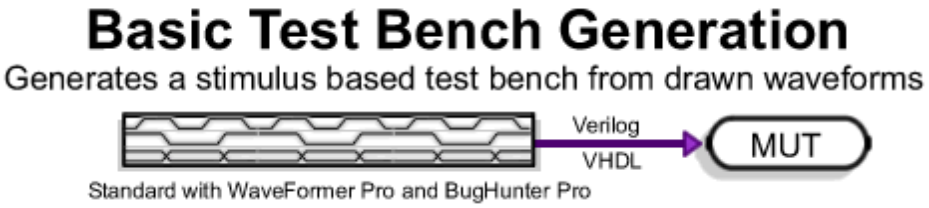
(TD) 6.17 Summary

This concludes the advanced simulation tutorial. Other simulation features not covered in this tutorial that you may wish to experiment with are: flip-flop timing characteristics (clock to output propagation delay and continuous setup and hold time checking) in the *Signal Properties* Dialog and the global simulation options in the **Options > Simulation Preferences Dialog**.

Copyright © 2014, SynaptiCAD Sales, Inc.

Test Bench Generation 1: VHDL-Verilog Stimulus

This tutorial describes how to generate Verilog and VHDL basic stimulus test benches using WaveFormer Pro, VeriLogger, Reactive Test Bench Option, and TestBencher Pro. It explores how different waveforms and state values in a timing diagram will affect the generation of the test bench code. It also explores the SynaptiCAD language-independent signal types which allow a single timing diagram to generate both VHDL and Verilog test benches. The generated test benches are normally used with external simulators, and with FPGA tools like Xilinix ISE, Altera Quartus, and Microsemi (Actel) Libero.



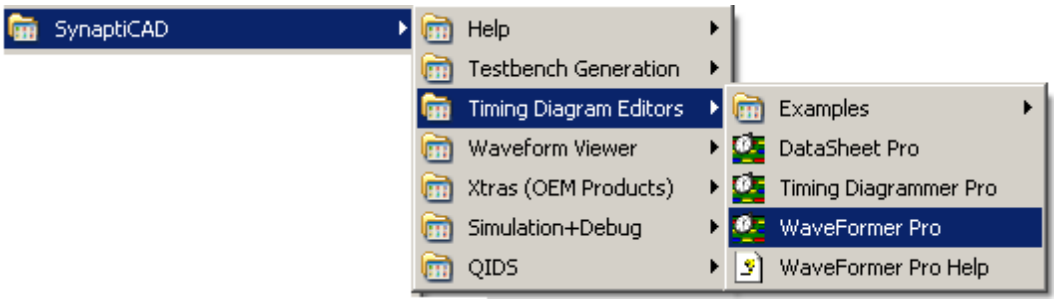
SynaptiCAD offers three levels of test bench generation. This tutorial demonstrates the basic level ***stimulus based test benches*** that are generated by the default configuration of WaveFormer Pro and BugHunter Pro with VeriLogger. Purchasing the add-on Reactive Test Bench Option allows users to create more advanced ***self-testing test benches*** which generate error reports and react to the model under test during simulation. This feature is demonstrated in the tutorial Test Bench Generation 2: Reactive Test Bench Option 122. The highest level of testbench generation is provided by TestBencher Pro, which allows a user to design ***bus functional models using multiple timing diagrams*** and a sequencer process to apply the diagrams. TestBencher can be added to BugHunter or purchased as a standalone product and is demonstrated in the tutorial Test Bench Generation 3: Test Bencher Pro Basic Tutorial 142.

(TBench) 1.1 Load the Tutorial Timing Diagram

This tutorial requires a full version license of WaveFormer Pro, BugHunter Pro, or Test Bencher Pro because you will need to generate the test bench and save files. Timing Diagrammer Pro cannot generate test benches. To obtain a temporary license for evaluation purposes, complete the form under the **Help > Request License** menu item and contact our sales department.

Run WaveFormer Pro, BugHunter Pro, or Test Bencher Pro:

- Run one of the above programs from the Start Menu.

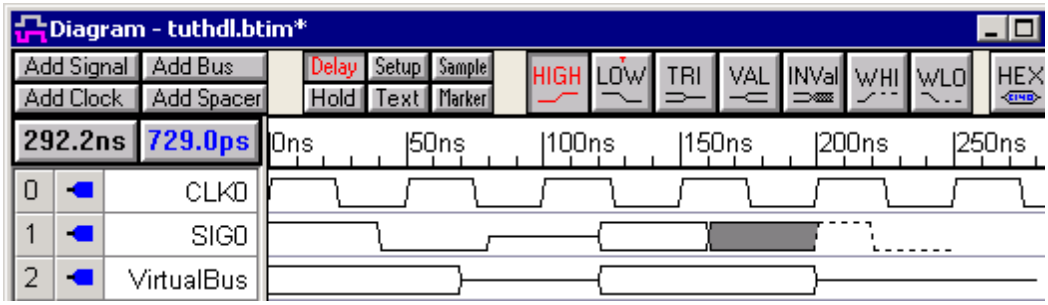


Load the starting Timing Diagram:

- Select the **File > Open** menu option and load the file **tuthdl.btim** from the **SynaptiCAD**

Copyright © 2014, SynaptiCAD Sales, Inc.

\Examples\TutorialFiles\AdvancedHDLStimulusGeneration directory.



The first signal, **CLK0**, is a clock with a period of 50ns. The second signal, **SIG0**, is a waveform that contains all of the graphical states available in WaveFormer Pro. The third signal, **VirtualBus**, is a waveform drawn with valid and tri-state segments that you will add values to in the next section. The blue icons pointing to the left mean that the signals are outputs of the test bench and will be exported when you generate the code.

Save the starting Timing Diagram:

- Select the **File > Save As** menu option and save the timing diagram as **test.btim** (this will keep the original file intact).

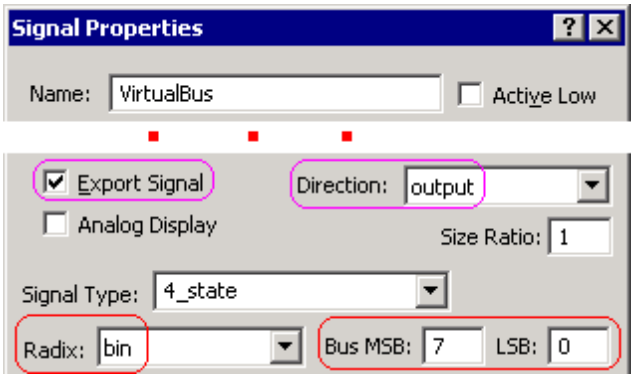
(TBench) 1.2 Hex and Binary State Values

SynaptiCAD's timing diagram editing environments use a language independent bus format for hexadecimal and binary bus values so that one diagram can generate a valid test benches for both VHDL and Verilog simulators. During test bench generation, single-bit signals generate code that matches the graphical state.

Buses (multi-bit signals) use one of two methods to generate code. First, if a segments value begins with a **'b** or **'h** then it is assumed that the number is a binary or hexadecimal number and the number will be translated to the appropriate VHDL or Verilog bus value (regardless of the radix of the signal). If the extended state value does not start with 'b or 'h then the value is written out as it was entered, without any translation. This means that if a state has a value of 10 and a radix of Hexadecimal, the effective value of the segment is 16 decimal. And if a state has a value of 10 with a radix of Binary, the effective value of the segment is 2 decimal.

Set the size and radix of the VirtualBus bus signal:

- Double click on the **VirtualBus** signal name to open the *Signal Properties* dialog.
- Make it an 8 bit bus by typing **7** into the **MSB** box and **0** into the **LSB** box.
- Set the **Radix** to **binary**.
- Also notice that **Export Signal** is checked so the signal will be exported and that it has a direction of **output** so it will drive the model under test.

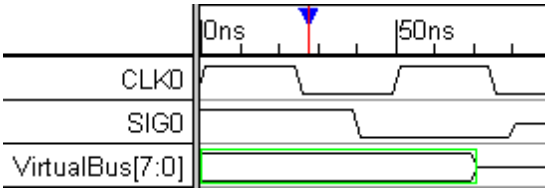


Copyright © 2014, SynaptiCAD Sales, Inc.

- Close the *Signal Properties* dialog.

Setting the values in a virtual bus waveform:

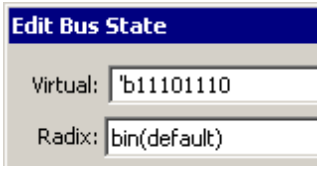
- **Select** the first waveform segment of *VirtualBus* by clicking on it. A selected segment has a box around it.



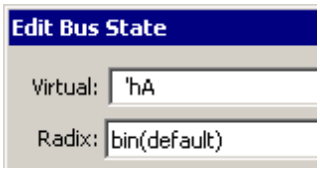
- Click on the **HEX** button at the top of the window to open the *Edit Bus State* dialog box. The *Edit Bus State* dialog box can also be opened by **double-clicking** on the selected segment.



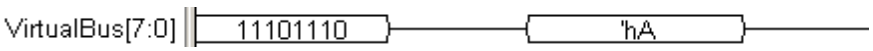
- Type **'b11101110** into the **Virtual** edit box (this is an 8-bit binary number).
- Press **ALT-N** (or press the **Next** button) two times to advance to the next valid segment



- Type **'hA** into the **Virtual** edit box (this is an 8-bit hexadecimal number equal to 00001010 in binary). The program automatically left pads missing bits with zeros.



- Press **OK** to close the *Edit Bus State* dialog. The waveform should look like the following.

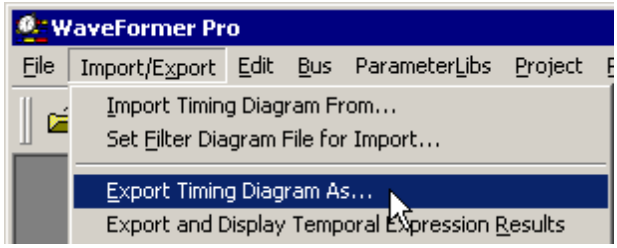


(TBench) 1.3 Export a Verilog Test Bench

Test benches can be generated by both the **File > Save Timing Diagram As** and the **Export > Export Timing Diagram As** menus. However the File menu always defaults to the SynaptiCAD BTIM format. Once you setup the Export menu to the settings for your favorite export format, it will default to that format. If you are a VHDL-only person you can skip this and go to the next section.

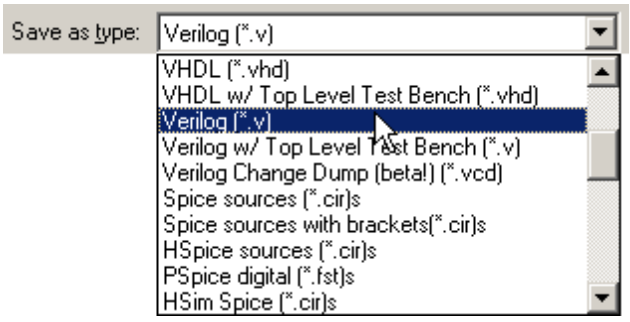
Generate the Verilog Test Bench:

- Choose the **Export > Export Timing Diagram As** menu option to open the *Export* dialog.

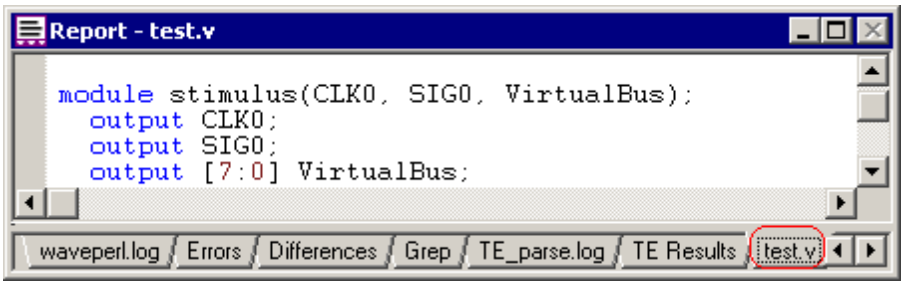


Copyright © 2014, SynaptiCAD Sales, Inc.

- In the **Save as Type** list box in the lower left corner of the dialog, choose the **Verilog (*.v)** script. This indicates that the timing diagram will be exported to a Verilog code file with a default file extension of ".v".



- Choose **test.v** as the file name and click the **Save** button to close the dialog. WaveFormer Pro will produce a Verilog file named test.v.
- The file **test.v** is automatically displayed in the Report window. If you cannot see the *Report* window, select **the Window > Report Window** menu option to bring the window to the top.

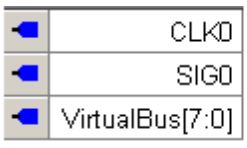


Compare the Code to the Diagram:

- At the top of the file there is a large comment section that describes how the code was generated, what the clocking domains are, and what program features were used to generated the code. In the basic testbench generation, all signals are in the Unclocked domain (all signals are delayed by time values). With the reactive test bench option, you can delay signals based on the occurrence of clock edges as well.

```
// Generated by WaveFormer Pro Version 12.30a at 9:6:20 on 6/9/2008
// Stimulator for stimulus

// Generation Settings:
//   Export type: Stimulus only (reactive export not enabled)
//                Delays, Samples, Markers, etc will not generate code.

// Clock Domains:

//   Unclocked
//   ---------
//     Signals:
//       SIG0
//       VirtualBus
```

- Notice that each of the output signals in the diagram are also outputs of the generated stimulus module. These signals will hook up to your model under test.




```
module stimulus(CLK0, SIG0, VirtualBus);
  output CLK0;
  output SIG0;
  output [7:0] VirtualBus;
```

Copyright © 2014, SynaptiCAD Sales, Inc.

- CLK0 is defined as a clock signal in the timing diagram. The program generates a loop to represent the clock (rather than a series of assignment statements for each edge). Also notice that all of the clock properties such as buffer delay and rise and fall jitter will generate proper code.

```
always
  begin
  @(posedge tb_status[0])
  offset = $bitstoreal( offset_bits );
  period = $bitstoreal( period_bits );
  duty   = $bitstoreal( duty_bits );
  minLH = $bitstoreal( minLH_bits );
  maxLH = $bitstoreal( maxLH_bits );
  minHL = $bitstoreal( minHL_bits );
  maxHL = $bitstoreal( maxHL_bits );
  jRise = $bitstoreal( jRise_bits );
  jFall = $bitstoreal( jFall_bits );
```

- Compare the signal waveforms to the generated code. Notice that the undefined valid state on SIG0 generates to 1'bx, but the defined valid states on Virtual bus generate out to the proper values.




```
initial
  begin
  SIG0_driver <= 1'b1;
  VirtualBus_driver <= 8'b11101110;
  end

task Unclocked;
  begin
  #40;
  SIG0_driver <= 1'b0;
  #30;
  VirtualBus_driver <= 8'hzz;
  #10;
  SIG0_driver <= 1'bz;
  #40;
  SIG0_driver <= 1'bx;
  VirtualBus_driver <= 8'h0A;
  #40;
  SIG0_driver <= 1'bx;
  #40;
  SIG0_driver <= 1'b1;
  VirtualBus_driver <= 8'hzz;
  #20;
  SIG0_driver <= 1'b0;
  #60.576;
  end
endtask
```

(TBench) 1.4 Signal Data Types and VHDL user defined types

SynaptiCAD's tools use a language independent signal type so that one timing diagram can generate code for both VHDL and Verilog test benches. Double click on a signal name to open the *Signal Properties* dialog and the **Signal Type** box shows the signal's type. Below is a table that shows the translation 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

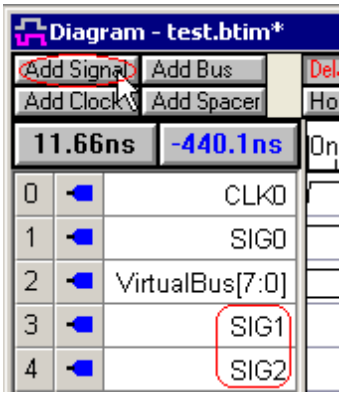
Copyright © 2014, SynaptiCAD Sales, Inc.

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

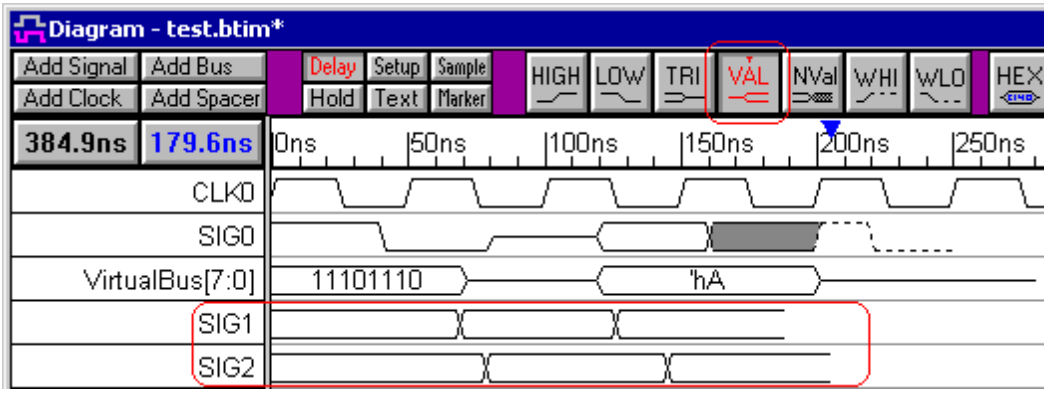
The **Signal Type** box also supports VHDL user-defined types that can be directly entered into the box. By default all signals are assumed to have a type of **std_logic** and a direction of **out** (CLK0, SIG0, and *VirtualBus* will use the defaults for this tutorial). In this section you will add SIG1 and SIG2 to demonstrate signals with a standard integer type and a user defined type.

Add SIG1 and SIG2

- Click on the **Add Signal** button two times to add two signals.



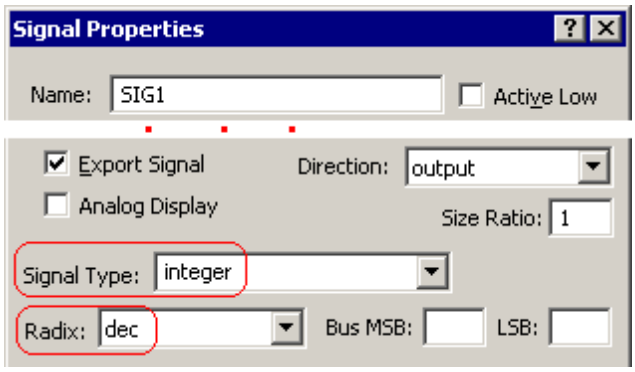
- Press the **VAL** button two times (not a double click). The first click selects Valid as the initial graphical state, and the second click selects Valid as the toggle state (as indicated by the red T). This causes the VAL button to stay selected when you draw waveform segments.
- Sketch some valid waveforms for **SIG1** and **SIG2** similar to those in the figure below.



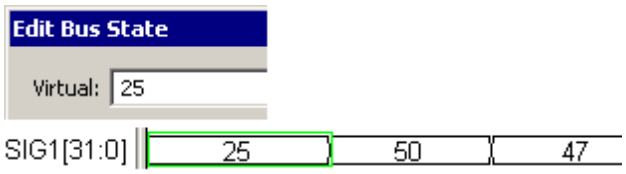
Copyright © 2014, SynaptiCAD Sales, Inc.

Change the type of SIG1 to integer and add values to the waveform:

- Double-click on the **SIG1** signal name to open the *Signal Properties* dialog.
- Choose **integer** from the **Signal Type** box.
- Choose **dec** from the **Radix** box to indicate that the values you will be entering into the virtual state are decimal values.
- Press **OK** to close the dialog.



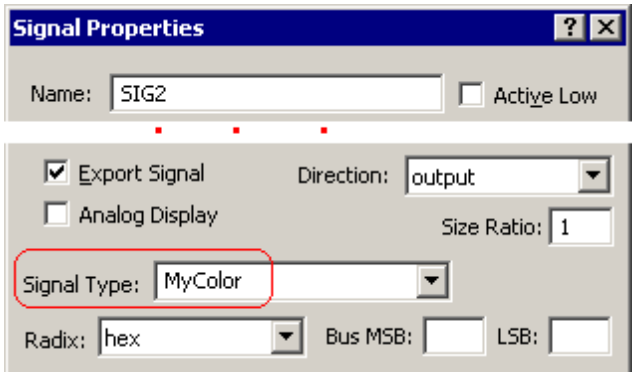
- Double-click on the **first** waveform segment on **SIG1** to open the *Edit Bus State* dialog.
- Enter **integer values** for each segment (we used **25**, **50**, **47**). Use the **Next** button to move between segments.



Change the type of SIG2 to MyColor and add values to the waveform:

Next we will add a user defined data type called MyColor. For a real VHDL simulation, the user defined type and the color value of the waveforms would have to match the code in the model under test. For instance, MyColor might be defined as: **enum MyColor ={RED, GREEN, BLUE, BLACK}** in the model under test code.

- Double-click on the **SIG2** signal name to open the *Signal Properties* dialog.
- Type in **MyColor** into the **Signal Type** box. *MyColor* is the name of the user defined type that we will use.
- Press **OK** to close the dialog.

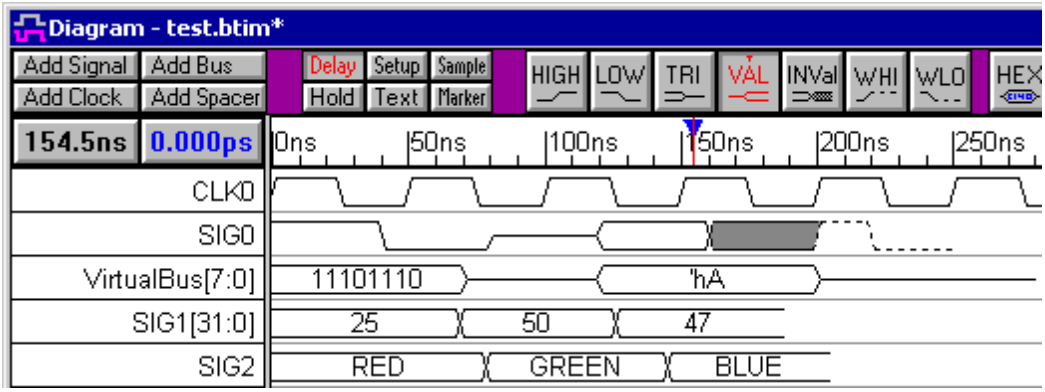


- Double-click on the **first** waveform segment on **SIG2** to open the *Edit Bus State* dialog.
- Enter some color value for each segment (we used RED, GREEN, BLUE).
- Click OK to close the dialog



- Your timing diagram should resemble the figure below.

Copyright © 2014, SynaptiCAD Sales, Inc.

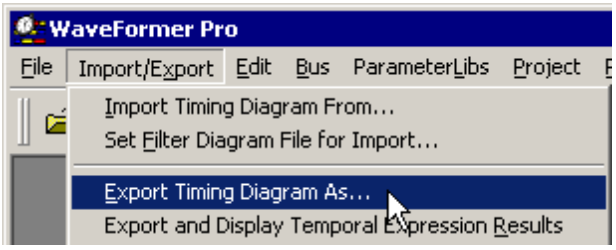


(TBench) 1.5. Export a VHDL Test Bench

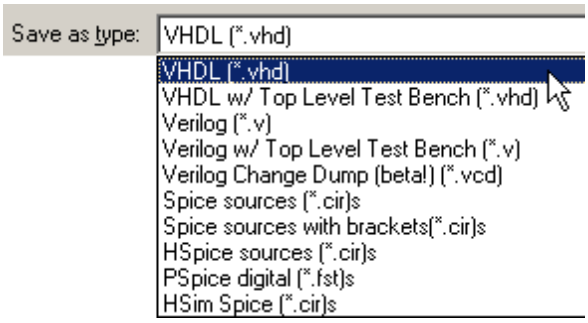
Next we will export to a VHDL test bench. If you are a Verilog only person, you can generate a Verilog test bench instead and investigate how the integer signal got exported to Verilog.

Generate the VHDL Test Bench:

- Choose the **Export > Export Timing Diagram As** menu option to open the *Export* dialog.

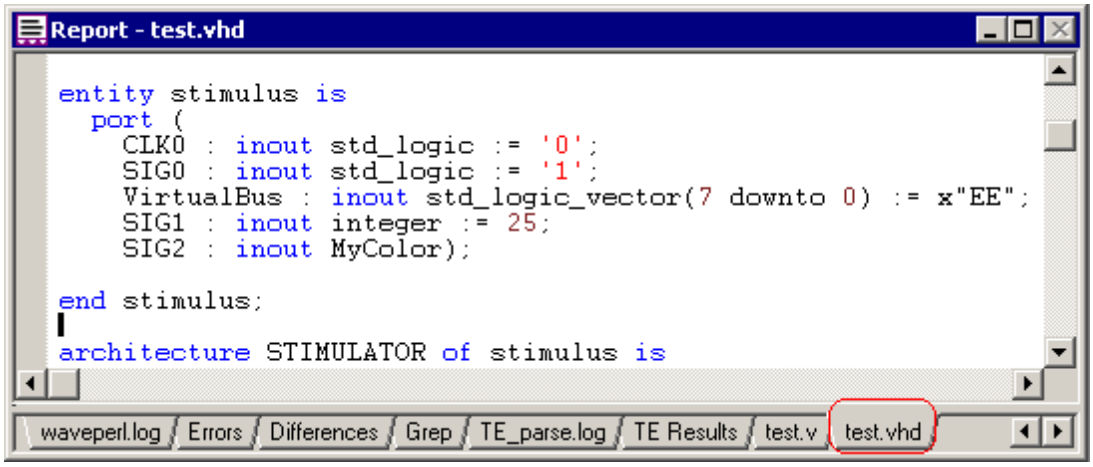


- In the **Save as Type** list box in the lower left corner of the dialog, choose the **VHDL (*.vhd)** script. This indicates that the timing diagram will be exported to a VHDL code file with a default file extension of ".vhd".



- Choose **test.vhd** as the file name and click the **Save** button to close the dialog. WaveFormer Pro will produce a VHDL file named test.vhd.
- The file **test.vhd** is automatically displayed in the Report window. If you cannot see the *Report* window, select **the Window > Report Window** menu option to bring the window to the top.

Copyright © 2014, SynaptiCAD Sales, Inc.



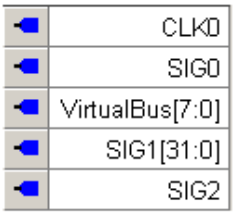
Compare the Code to the Diagram:

View the file **test.vhd** inside the *Report* window. Notice the entity and architecture structures and the types of all the signals. CLK0 uses a while loop to calculate its value. **SIG0** shows how the graphical states are exported. **VirtualBus** is defined as an 8-bit logic vector. SIG1's values are exported as integers. SIG2's values are exported as RED, GREEN, and BLUE.

- At the top of the file there is a large comment section that describes how the code was generated, what the clocking domains are, and what program features were used to generated the code. In the basic testbench generation, all signals are in the Unclocked domain (all signals are delayed by time values). With the reactive test bench option, you can delay signals based on the occurrence of clock edges as well.

```
-- Generated by WaveFormer Pro Version 12.30a at 13:44:0 on 6/9/2008
-- Stimulator for stimulus

-- Generation Settings:
--   Export type: Stimulus only (reactive export not enabled)
--                Delays, Samples, Markers, etc will not generate code.

-- Clock Domains:

--   Unclocked
--   ---------
--     Signals:
--       SIG0
--       VirtualBus
--       SIG1
--       SIG2
```

- Notice that each of the output signals in the diagram are also inout of the stimulus entity. These signals will hook up to your model under test.



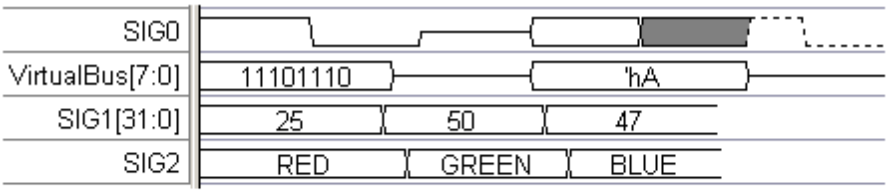

```
entity stimulus is
  port (
    CLK0 : inout std_logic := '0';
    SIG0 : inout std_logic := '1';
    VirtualBus : inout std_logic_vector(7 downto 0) := x"EE";
    SIG1 : inout integer := 25;
    SIG2 : inout MyColor);

end stimulus;
```

Copyright © 2014, SynaptiCAD Sales, Inc.

- CLK0 is defined as a clock signal in the timing diagram. The program generates a loop to represent the clock (rather than a series of assignment statements for each edge). Also notice that all of the clock properties such as buffer delay and rise and fall jitter will generate proper code.

```
-- Clock Instantiation
tb_CLK0 : entity syncad_vhdl_lib.tb_clock_minmax
  generic map (name => "tb_CLK0",
               initialize => true,
               state1 => '1',
               state2 => '0')
  port map (tb_status,
            CLK0,
            CLK0_MinLH,
            CLK0_MaxLH,
            CLK0_MinHL,
            CLK0_MaxHL,
            CLK0_Offset,
            CLK0_Period,
            CLK0_Duty,
            CLK0_JRise,
            CLK0_JFall);
```

- Compare the signal waveforms to the generated code. Notice that the undefined valid state on SIG0 generates to 1'bx, but the defined valid states on Virtual bus generate out to the proper values.

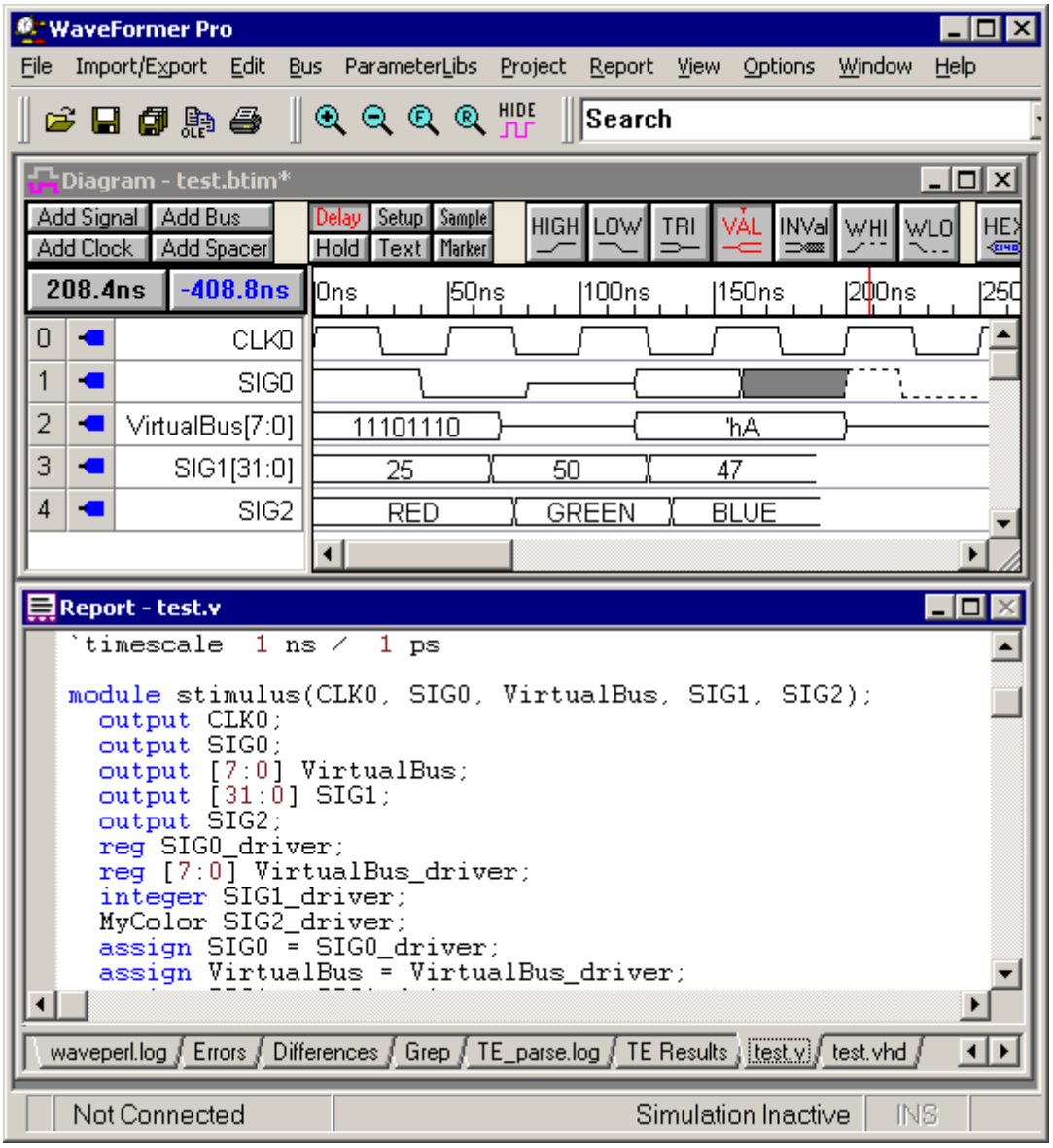



```
-- Sequence: Unclocked
Unclocked : process
begin
  wait for 40 ns;
  SIG0 <= '0';
  wait for 26 ns;
  SIG1 <= 50;
  wait for 4 ns;
  VirtualBus <= "ZZZZZZZZ";
  wait for 5 ns;
  SIG2 <= GREEN;
  wait for 5 ns;
  SIG0 <= 'Z';
  wait for 40 ns;
  SIG0 <= 'X';
  VirtualBus <= x"0A";
  wait for 5 ns;
  SIG1 <= 47;
  wait for 9 ns;
  SIG2 <= BLUE;
  wait for 26 ns;
  SIG0 <= 'X';
  wait for 40 ns;
  SIG0 <= 'H';
  VirtualBus <= "ZZZZZZZZ";
  wait for 20 ns;
  SIG0 <= 'L';
  wait for 60.576 ns;
  wait;
end process;
end STIMULATOR;
```

Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 1.6 Summary of VHDL-Verilog Stimulus Tutorial

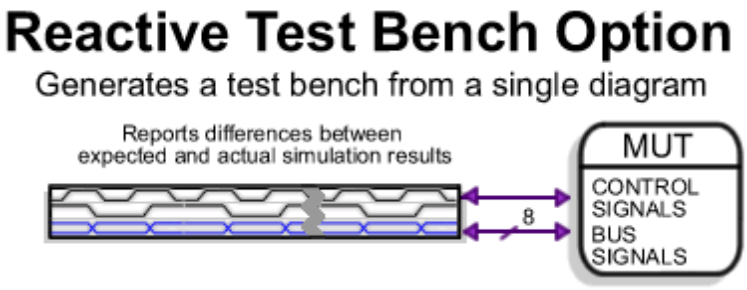
Congratulations, you have now completed the VHDL-Verilog Stimulus Tutorial. You have generated both a VHDL and a Verilog test bench using the **Export > Export As** menu function. You have investigated the language independent waveform state values and the signal data types. And you have used the Report window to view the generated code. For more information on the basic VHDL/Verilog generation please refer to the Timing Diagram Editor Manual Chapter 4: Simulated Signals and VHDL/ Verilog Export.



Copyright © 2014, SynaptiCAD Sales, Inc.

Test Bench Generation 2: Reactive Test Bench Option

The Reactive Test Bench Generation Option can be added to WaveFormer Pro, DataSheet Pro, VeriLogger, and BugHunter Pro. The features are included in TestBencher Pro, so it is also a good introduction to creating a single timing transaction in TestBencher Pro. With Reactive Test Bench Generation, users can draw "expected" waveforms on the MUT output ports and add "samples" to the waveforms to test for specific state values. During simulation, the code generated by the samples would watch the output from the model under test and compare it to the drawn waveform states. Samples can perform a variety of functions such as pausing the simulation to debug a problem, reporting errors and warnings, user-defined actions, and triggering other samples. The generated test benches are normally used with external simulators, and with FPGA tools like Xilinix ISE, Altera Quartus, and Microsemi (Actel) Libero.



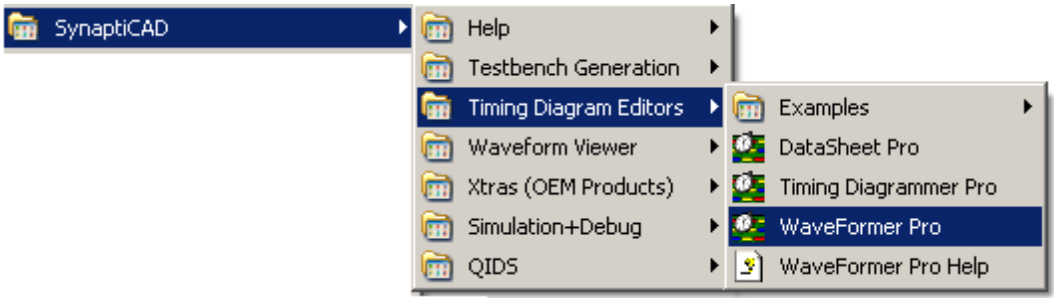
(TBench) 2.1 Run Program with Reactive Test Bench Option

This tutorial requires a Reactive Test Bench Generation license plus a license for one of the SynaptiCAD products that supports this feature. To obtain a temporary license for evaluation purposes, complete the form under the **Help > Request License** menu item and contact our sales department.

Run WaveFormer Pro, BugHunter Pro, WaveFormer Lite (Libero) or Test Bencher Pro:

If you are using the Actel Libero tool set, skip this step because you will be launching WaveFormer Lite from within Libero as described in the next section.

- Run one of the above programs from the Start Menu.



(TBench) 2.2 Create a Project to hold the MUT

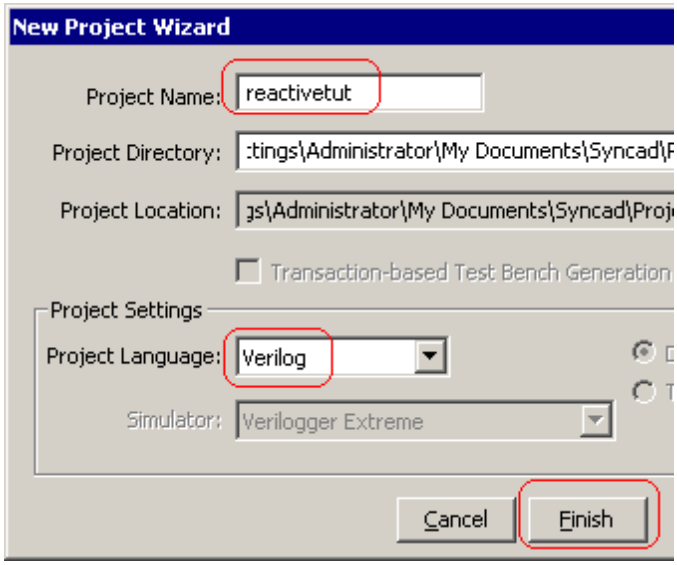
First, create a project file to hold the model under test. This will allow WaveFormer to extract the information necessary to instantiate the model under test inside the testbench and extract the input and output ports from the model under test. The model under test in this tutorial is a Verilog file, so we will be generating a Verilog test bench, but the diagram and the design flow is the same for a

Copyright © 2014, SynaptiCAD Sales, Inc.

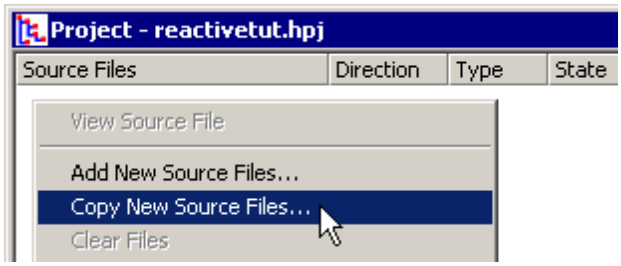
VHDL test bench and you can generate a VHDL test bench from the extracted signal information.

Non-Libero users create a SynaptiCAD Project and add the Model Under Test:

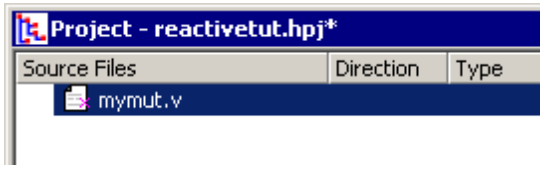
- Choose **Project > New Project** menu function to open the *New Project Wizard* dialog.
- In the **Project Name** box, type in **reactivetut** as the name of the project. This will become both the name of the project and the directory where the project and associated files are stored.
- Choose **Verilog** as the **Project Language**.
- Click the **Finish** button to create the project and close the dialog.



- In the **Project** window, under the **Source Files** section, right click and choose **Copy New Source Files** from the context menu. The picture shows WaveFormer's interface. BugHunter and TestBencher projects have a **User Source Files** folder that you can right click on. This will open a file open dialog.



- Select **mymut.v** from the **SynaptiCAD\Examples\TutorialFiles\ReactiveTestBench** directory and close the dialog to copy the file to the project directory.



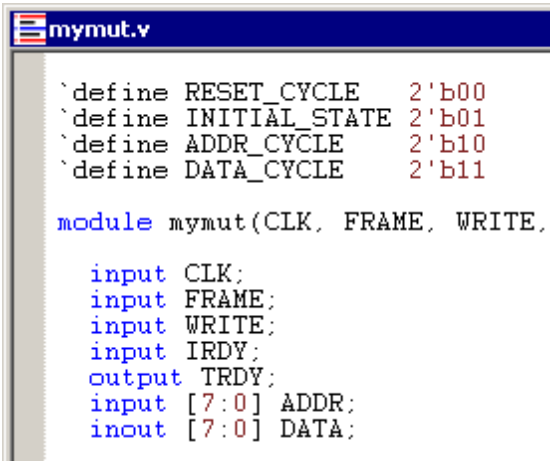
Actel Libero users only:

Create a project inside Libero following the steps in the Libero documentation, add the source file **mymut.v** from the **SynaptiCAD\Tutorials\ Reactive Test Bench** directory**,** and launch WaveFormer Lite. This will automatically create a WaveFormer project file and add your source files to that project.

Copyright © 2014, SynaptiCAD Sales, Inc.

Investigate the Model Under Test

- In the Project window, double click on the **mymut.v** to open an Editor window which displays the model under test source code.



We will use a simplified version of a PCI slave device as the model to be tested. No experience with PCI is required to perform and understand this tutorial. There is no arbitration, the MUT responds to all addresses, and the only valid commands are single reads and writes. It contains a memory that can be written to and read from and has the following ports (all control signals are active low):

CLK (input): the mymut device is clocked on the positive edge

FRAME (input): indicates start of transaction.

WRITE (input): indicates write transaction.

IRDY (input): stands for **initiator ready**. Indicates when the master device is ready for transaction to complete (the master will be the test bench in this case).

TRDY (output): stands for **target ready**. During a write, this indicates that the MUT has finished writing data to it's memory. During a read, this indicates that the MUT has read the data from memory and put it on the DATA bus.

ADDR (output): Address to write to or read from.

DATA (inout): Data to write to memory or data that is read from memory.

Each transaction consists of an address cycle and data cycle. During the address cycle, the **WRITE** and **ADDR** signals must be valid. During a write data cycle, the **DATA** signal must be valid before **IRDY** is asserted. Then the MUT indicates that it is finished storing the data by asserting **TRDY**. During a read data cycle, the MUT must drive **DATA** before asserting **TRDY**. Then, the master asserts **IRDY** when it is finished reading the data. Once **IRDY** or **TRDY** is asserted, they must remain asserted until the transaction is finished which is indicated by the de-assertion of **FRAME**.

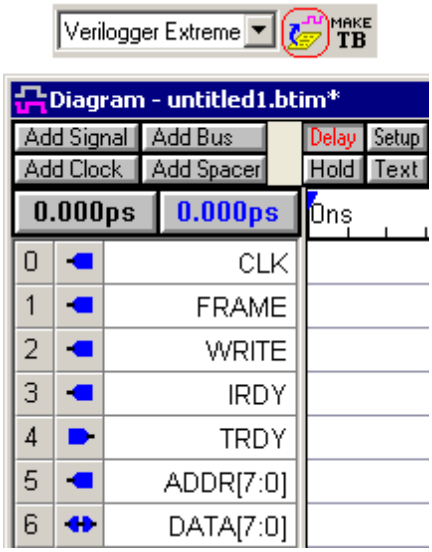
(TBench) 2.3 Extract Signal Names and setup the Clock

Next we will extract the signal names, types, and size from the port information contained in the mymut.v file.

Copyright © 2014, SynaptiCAD Sales, Inc.

Extract MUT ports into Diagram

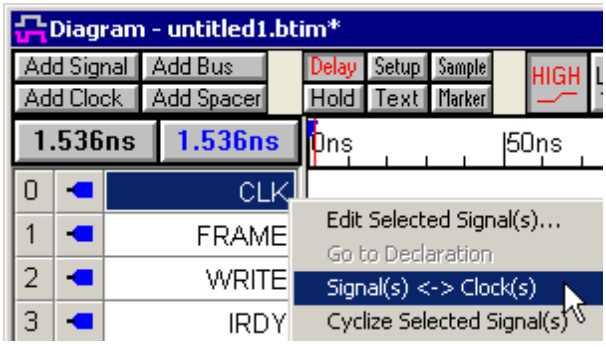
- Press the **Extract MUT ports into Diagram** button on the main window button bar to extract the top level ports from **mymut.v** and dump them into a timing diagram window. Notice that these ports match the signals in the mymut.v code



Convert the CLK signal into a SynaptiCAD Clock Signal:

The Extract MUT Ports function imports all the port signals into SynaptiCAD Signals. However, we would like the clock to draw itself based on a frequency. To do this we need to covert the CLK signal into a CLK clock.

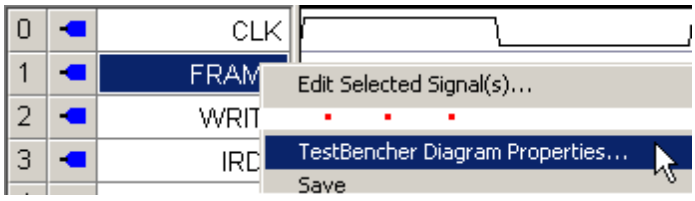
- Right-click on the CLK label name and select **Signal <-> Clock** from the context menu. This will convert the signal to a clock, and draw a clock waveform with a default frequency of **10MHz**.
- The default clock values are fine for this tutorial, but if you want to look at the Clock Properties, double click on the clock waveform to open the *Edit Clock Parameters* dialog. Close it when you are done.



Set the default Clock Signal to make a cycle based test bench

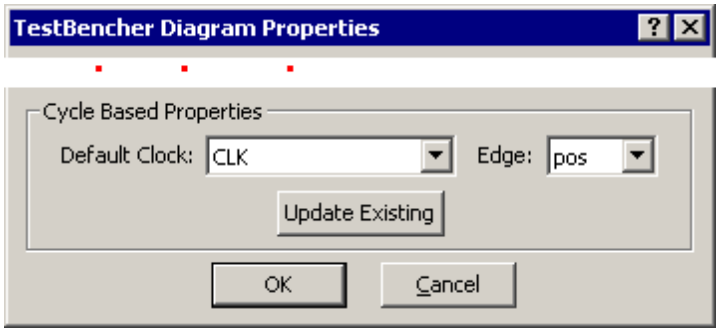
Next set the **Clock** signal and **Edge** for all of the signals in the diagram so that the test bench will be cycle-based instead of time-based (this means the test bench stimulus will change after waiting on clock transitions instead of time delays).

- Right-click on a signal name and select **TestBencher Diagram Properties** context menu to open a dialog of the same name.

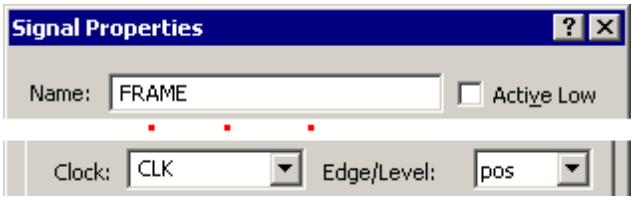


Copyright © 2014, SynaptiCAD Sales, Inc.

- In the **Default Clock** drop down, select **CLK** as the default clocking signal to use and set the *Edge* dropdown as **pos** to specify positive-edge clocking.
- Click the **Update Existing** button to set the clocking signal for existing signals. Press **OK** to close the dialog.



- That dialog set the clock signal for every normal signal in the diagram. This could have been done individually by double clicking on each signal name to open the *Signal Properties* dialog and setting the controls there. The individual settings are used when you have multiple clocking domains.



Examine the difference between a cycle-based and time-based test bench:

Following is a code example of the difference between a cycle-based and time-based test bench. Both of these code segments were exported from the diagram you will be drawing in the next step. Signals with the same clocking signal and edge type will be driven by a common process in the generated code. We call a process of this type a Clocked Sequence. All the unclocked signals are driven by the Unclocked Sequence. This means that when a diagram contains signals with different clocking signals, a separate sequence process will be created for each clocking signal/edge type. Clocked Sequences are named based on the clocking signal and the edge type, so for this example, the clocked sequence that contains the code on the right will be called **CLK_pos**.

Time Based Code

```
#137;
FRAME_driver <= 1'b0
#3;
WRITE_driver <= 1'b0
ADDR_driver <= 8'h00
#100;
WRITE_driver <= 1'b1
IRDY_driver <= 1'b0;
ADDR_driver <= 8'hxx
DATA_driver <= 8'hAA
#100;
```

Cycle-based Code

```
repeat (2)
begin
  @(posedge CLK);
end
FRAME_driver <= 1'b0;

WRITE_driver <= 1'b0;
ADDR_driver <= 8'h00;
@(posedge CLK);
WRITE_driver <= 1'b1;
IRDY_driver <= 1'b0;
ADDR_driver <= 8'hxx;
DATA_driver <= 8'hAA;
@(posedge CLK);
FRAME_driver <= 1'b1;
```

Copyright © 2014, SynaptiCAD Sales, Inc.

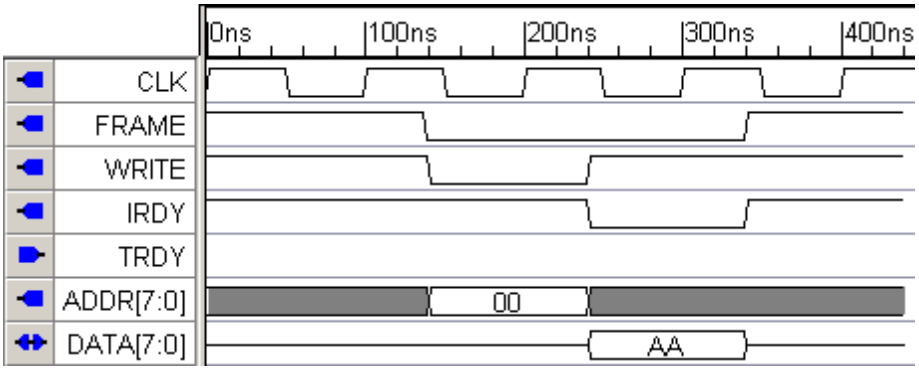
```
FRAME_driver <= 1'b1                IRDY_driver <= 1'b1;
IRDY_driver <= 1'b1;                DATA_driver <= 8'hzz;
DATA_driver <= 8'hzz                @(posedge CLK);
#101;
```

(TBench) 2.4 Draw Stimulus Waveforms and Export Test Bench

First we will draw a write transaction, and export it as a simple stimulus based testbench. This transaction ignores the **TRDY** signal (an input to the testbench) and doesn't verify that the data was actually written successfully to the MUT. We will add that functionality in the next step.

Draw or Load the write cycle timing diagram:

- Quickly sketch the waveforms into diagram that we have been working with. If you have trouble drawing the waveforms, please review the Basic Drawing and Timing Analysis Tutorial 10 before continuing with the rest of this tutorial.

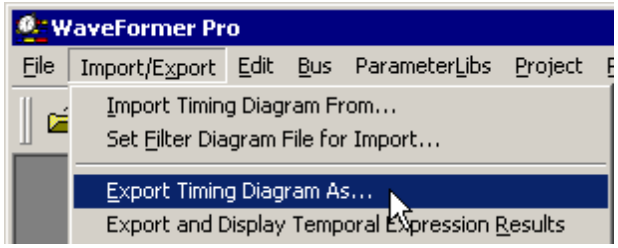


- Or load the completed write cycle timing diagram **draw_single_write.btim** located in the **SynaptiCAD\Examples\TutorialFiles\ReactiveTestBench\Completed Diagrams** directory.

Export a Verilog or VHDL test bench:

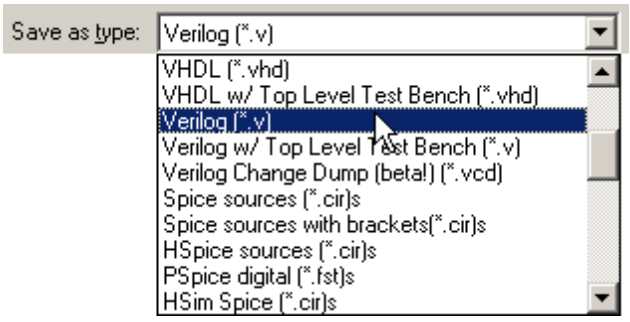
As you proceed through this tutorial, you will periodically generate the test bench to see how the generated code changes as the timing diagram is edited.

- Choose the **Export > Export Timing Diagram As** menu option to open the *Export* dialog.

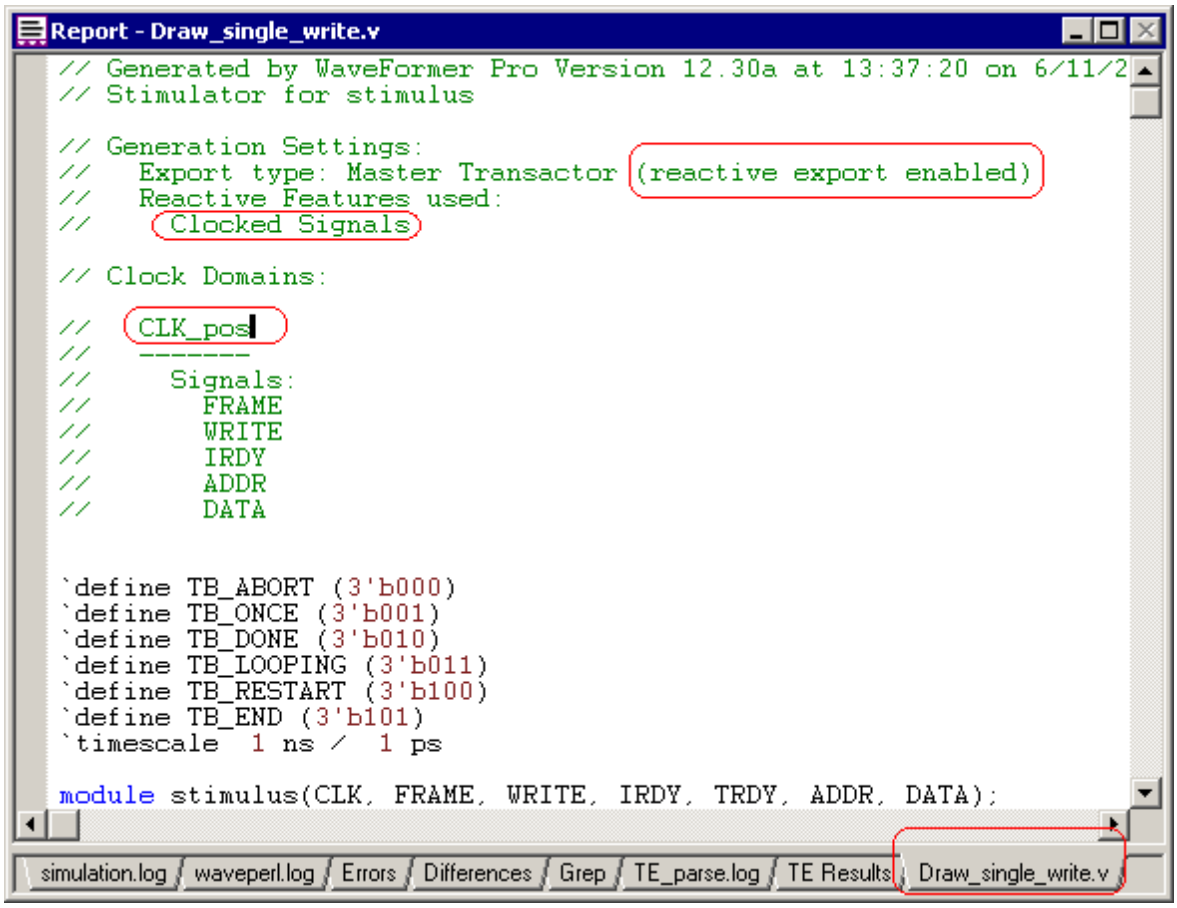


Copyright © 2014, SynaptiCAD Sales, Inc.

- In the **Save as Type** list box in the lower left corner of the dialog, choose either the **Verilog** or **Verilog w/ Top Level Test Bench**. The Verilog script just generates the test bench code. The Top Level Test Bench also includes an instantiation of the model under test.



- Choose **draw_single_write.v** as the file name and click the **Save** button to close the dialog and generated a test bench called draw_single_write.v.
- The file **draw_single_write.v** is automatically displayed in the Report window. If you cannot see the *Report* window, select **the Window > Report Window** menu option to bring the window to the top.



- View the generated code in the **Report** window. Notice that the comment at the top of the file shows that Reactive Export is enabled (if this is missing contact SynaptiCAD and get a license for this feature).
- Also notice that code also shows that all of the signals are in the **CLK_pos** clock domain. We set this in the previous section when we set the clocking signal and edge.
- Scroll through the code and quickly compare the code to the drawn waveforms.

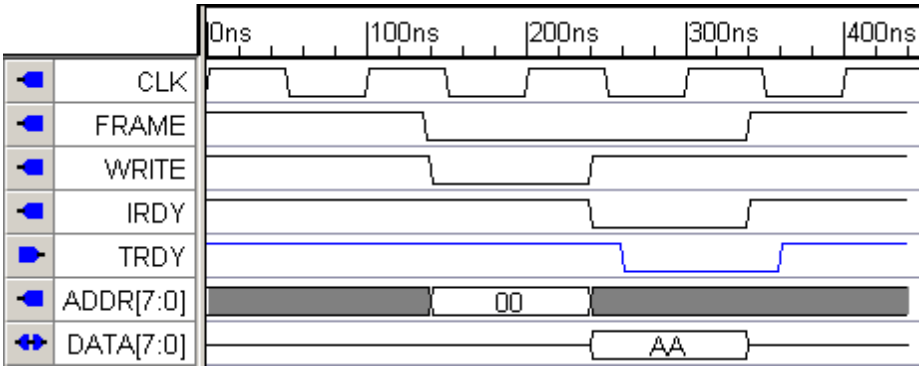
Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 2.5 Draw Expected Waveform and Wait for the Assertion

In this step we will sketch an expected TRDY waveform that the model under test should produce when it successfully completes the write cycle. We also want the test bench to pause until TRDY goes low before driving the rest of the stimulus. There are two different ways to create the pause.The first method uses a **Sample** which will pause until either the assertion happens or the sample timeouts. The other method uses the **Sensitive Edge** feature and will wait indefinitely for **TRDY** to assert. Both methods are explained below.

Draw the Expected Waveform for the TRDY signal

- First draw the expected **TRDY** waveform shown below. Notice that the waveform is blue because it is an input to the diagram, so the data shown is predicted data, not data to be driven. The direction of the signal was imported during the extract ports from mut step and is shown by the blue icon pointing to the right.



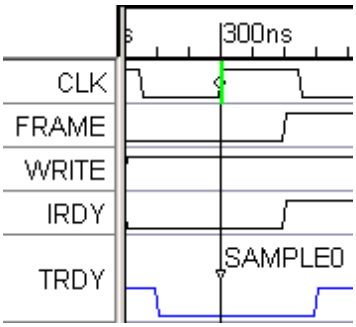
Method 1) Wait with a Timeout Using a Sample

This first method uses a graphical sample to wait on an incoming edge. The sample can also specify a timeout so the test bench does not hang-up during an error condition.

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

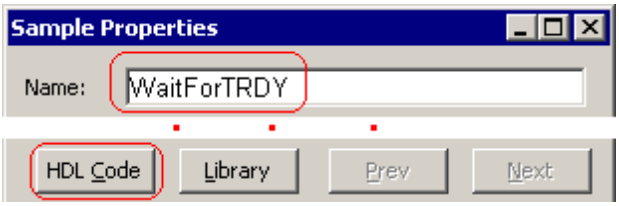


- **Left-click** on the rising edge of **CLK** at **300ns** to select the edge, then **right-click** on **TRDY** at **300ns** (or slightly later than 300 ns). This will add a sample to the diagram.
- Double-click on the new sample's name to open the *Sample Properties* dialog.

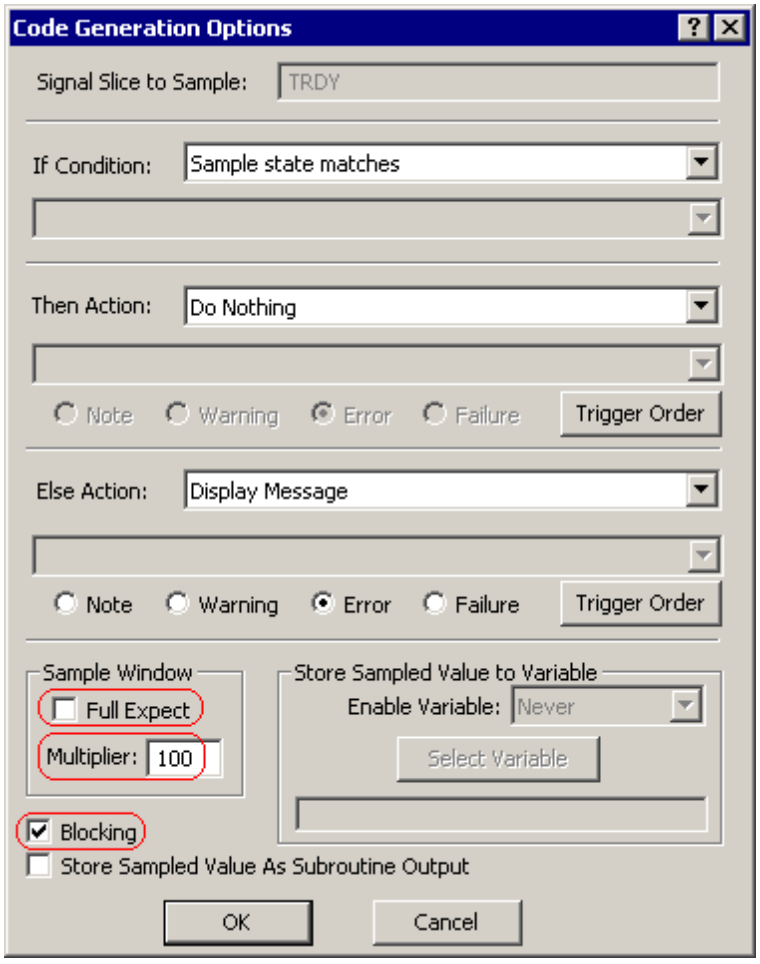


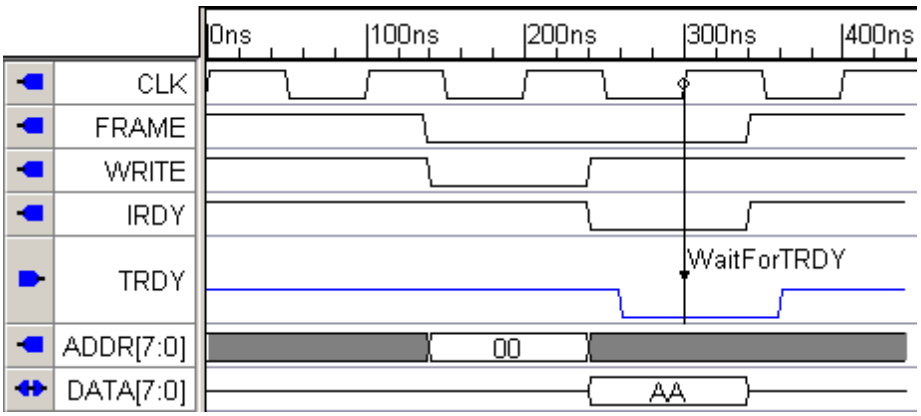
Copyright © 2014, SynaptiCAD Sales, Inc.

- Change the name to **WaitForTRDY,** then click on the **HDL Code** button in the *Sample Properties* dialog to open the *Code Generation Options* dialog.



- This dialog controls the code that the Sample Generates. Take a look at each of the controls.
- Uncheck the **Full Expect** check box to indicate TRDY only has to assert at some time during the sample's execution.
- Specify **100** for the *Multiplier* to make the sample wait for up to 100 cycles of the clock since TRDY is a clocked signal.
- Check the **Blocking** check box so that the sample blocks the transaction until the sample finishes.
- Close both dialogs.



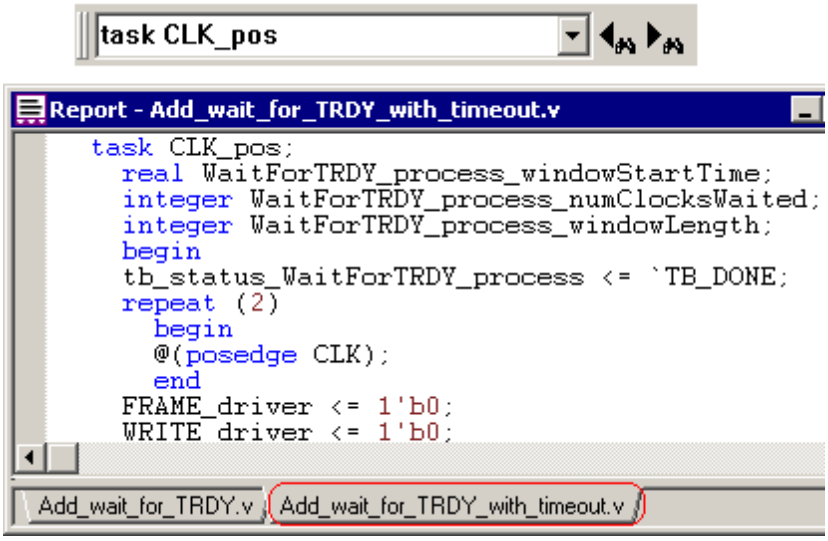


- Export the the test bench using the **Export > Export Timing Diagram** as menu as shown in

Copyright © 2014, SynaptiCAD Sales, Inc.

Section 2.4.

- Click into the Report Tab that contains the generated code to select the window.
- Then use the Search box on the button bar to locate the CLK_pos sequence (in verilog, search for "**task CLK_pos**", in VHDL search for "**CLK_pos :**").

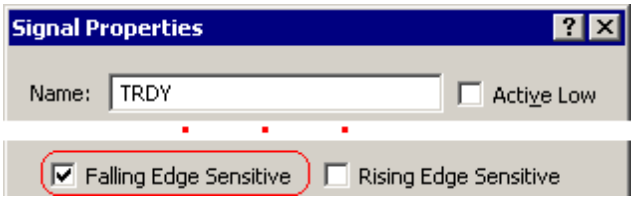


- Then search down for "**WaitForTRDY**" to see the code generated for the sample.
- This Method demonstrated a way to make the signal wait on a single event with an optional timeout. Click on the sample **WaitForTRDY** to select it and press the DELETE key to remove the sample so that we can demonstrate a different method in the next section.

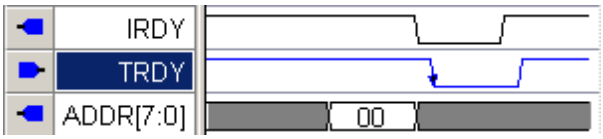
METHOD 2) Wait Indefinitely Using Sensitive Edges

A signal can be set to have sensitive edges, so that the test bench will wait on every falling or rising edge that the model under test generates. Here we will edit TRDY's properties so that the test bench will wait until TRDY has a falling edge before continuing to supply stimulus to the model under test.

- Double-click on **TRDY** to open the *Signal Properties* dialog.
- Check the **Falling Edge Sensitive** check box, and press **OK** to close the dialog.



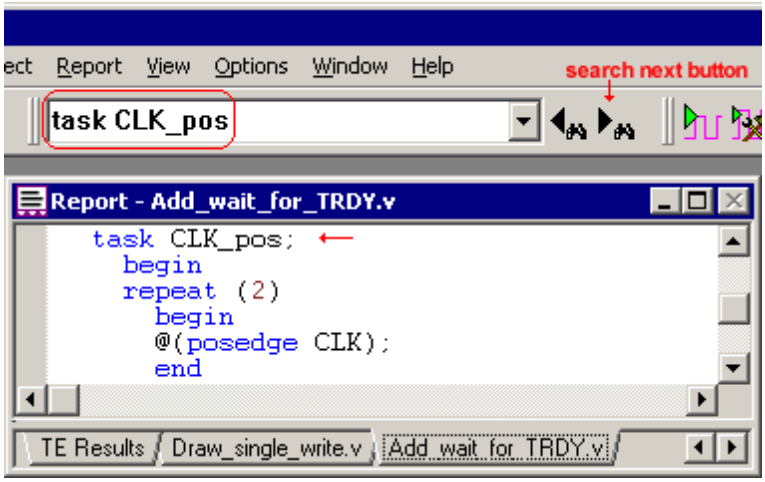
- Notice that TRDY now has an arrow drawn on the falling edge, to indicate that it is falling edge sensitive.



- Make sure that the falling edge of TRDY is drawn after the falling edge of IRDY, otherwise the test bench will wait for TRDY to assert before asserting IRDY.
- Export the the test bench using the **Export > Export Timing Diagram** as menu as shown in Section 2.4.
- Click into the Report Window Tab that contains the generated code to select the window.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Then use the Search box on the button bar to locate the CLK_pos sequence (in verilog, search for "**task CLK_pos**", in VHDL search for "**CLK_pos :**").
- Then search down for **"Sensitive"** to view the code generated for the sensitive edge. You should see code similar to the following:



VHDL:
```
-- Sensitive Falling Edge on signal: TRDY
  wait until falling_edge(TRDY) or (tb_DgmAborted);
```

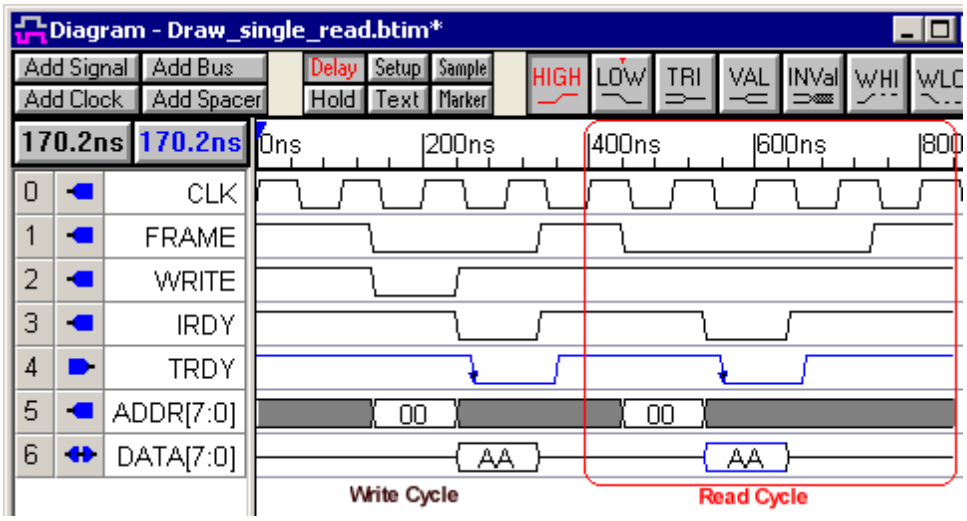
Verilog:
```
// Sensitive Falling Edge on signal: TRDY
   @(negedge TRDY);
```

(TBench) 2.6 Draw a Read Cycle and Verify the read

In this section, we will draw a read cycle and define the bidirectional segment on the DATA signal.

Draw the Waveforms for the Read cycle

- Either load the completed read cycle timing diagram **draw_single_read.btim** located in the **SynaptiCAD\Examples\TutorialFiles\ReactiveTestBench\Completed Diagrams** directory.
- Or draw the waveforms for the read which starts at 400ns. (See the instrutions below for making the multi-colored Data signal).



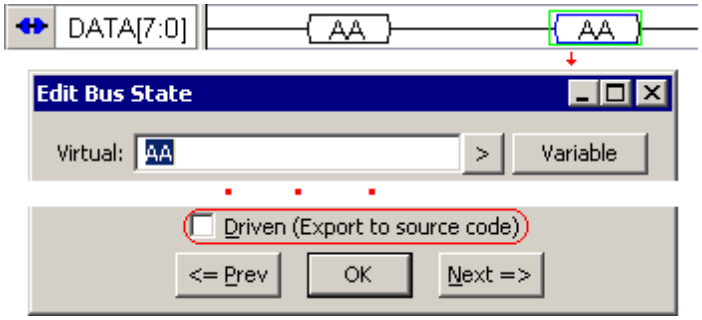
Draw the Waveforms for the Read cycle:

To avoid bus contention, the test bench must not drive the DATA bus during the read cycle, because the MUT will be driving that bus. Since the DATA bus is a bi-directional signal, you can specify which

Copyright © 2014, SynaptiCAD Sales, Inc.

parts of the waveform are driven by the test bench and which are not. One-way to do this is to draw a tri-state waveform. However, in this case we need to specify the expected data on the bus, so we will have to disable the drive on the expected value segments.

- Double-click on the waveform segment of **DATA** that happens during the read cycle to open the *Edit Bus State* dialog.
- Uncheck the **Driven** check box and click **OK**.



- Notice that the segment will be drawn in blue now, indicating that the **DATA** signal will *not* be driven by the test bench during this time period (just like the entire **TRDY** signal).

(TBench) 2.7 Add a Sample to Verify Data Read from MUT

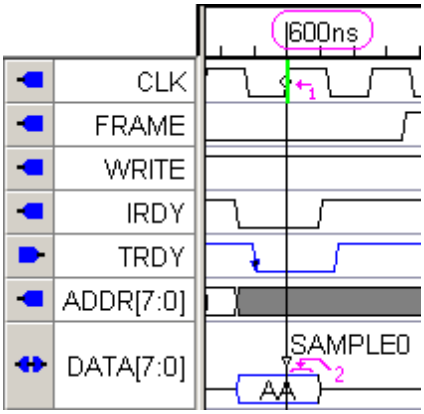
Samples can be used to verify data that is generated by the model under test during simulation. Here we will add a Sample to the Data bus to verify that the read cycle worked correctly.

Add a Sample to verify the read cycle:

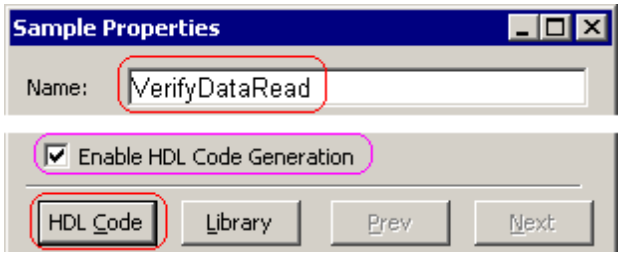
- Press the **Sample** button so that right clicks will add delays.



- **Left-click** on the positive **CLK** edge at **600ns**, then ***right-click*** on the **DATA** segment directly below it (slightly to the right). This will place a Sample that will trigger at that clock edge and verify that the data read from the MUT is what we expect (indicated by the waveform drawn under the Sample).

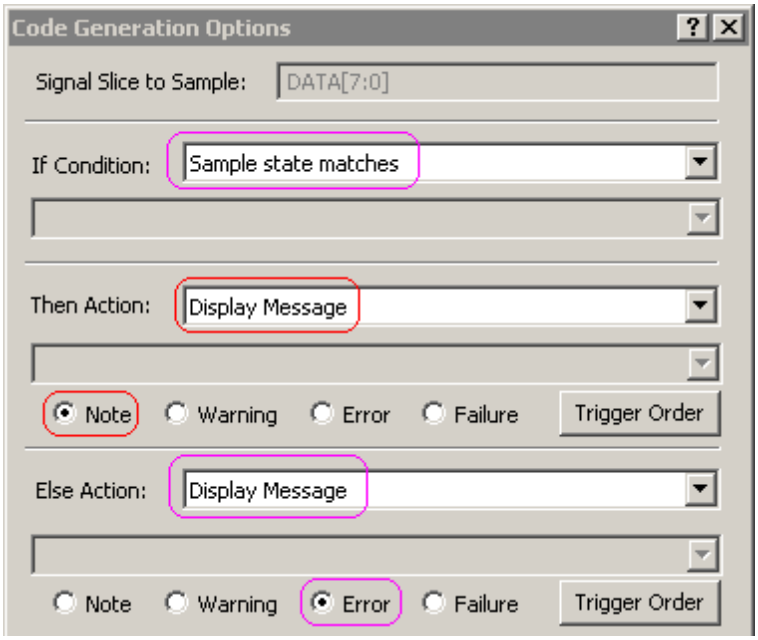


- Double-click on the Sample name to open the *Sample Properties* dialog.
- Change the sample name to **VerifyDataRead**
- Press the **HDL Code** button to open the *Code Generation Options* dialog.

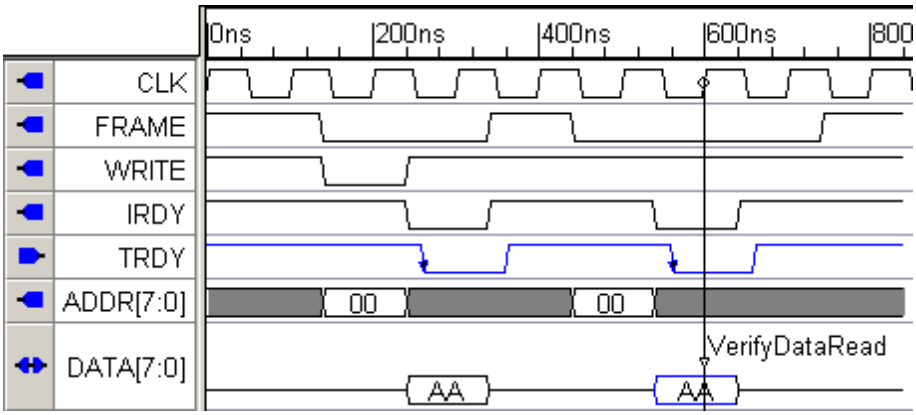


Copyright © 2014, SynaptiCAD Sales, Inc.

- The **If condition** should be set to ***Sample State Matches***, which is the default behavior for a sample.
- Set the **Then Action**, to ***Display Message*** so that each time the sample passes it will generate a log messages stating that it passed. And give it a warning level of **Note**.
- The **Else Action** should be set to ***Display Message*** so that each time the sample fails it will generate a log message stating the ***error***.
- Press **OK** to close both dialogs.

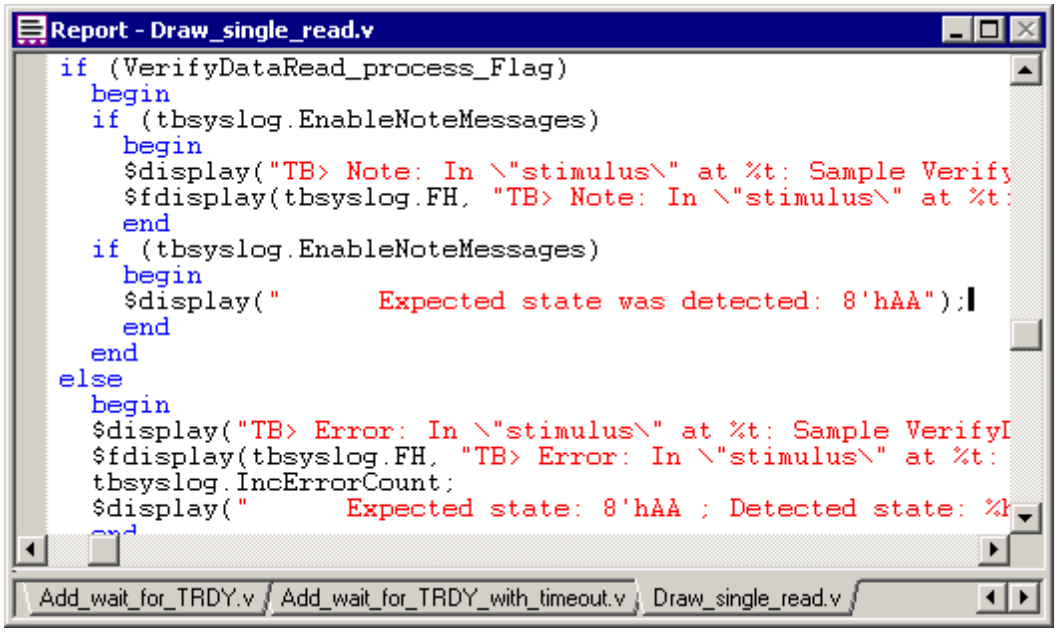


Here is what the diagram should look like after adding the Sample:



To see the generated code, export the test bench and view the code in the Report tab that contains the generated code. Locate the CLK_pos sequence, then search down for **VerifyDataRead** to see the code generated for the sample.

Copyright © 2014, SynaptiCAD Sales, Inc.



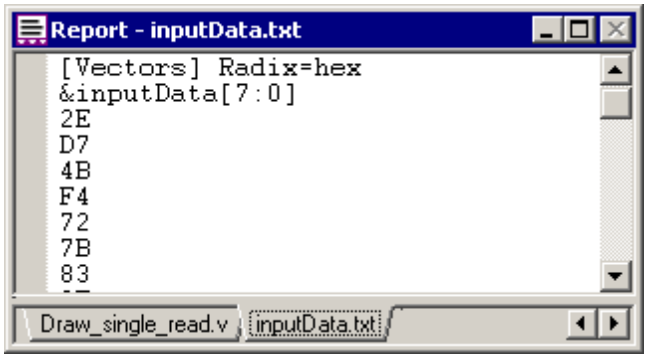
(TBench) 2.8 Drive Waveform Values using a File

This step will use an ASCII input file to drive the **DATA** bus during the write cycle so that we can model a memory array. The basic idea is to create a user-defined array variable that is initialized from a file containing the values stored in the memory, then drive the DATA bus using the values of this array (using the current address value as the index for the array).

View the Input File:

Waveform data values can be driven from a file that is in SynaptiCAD's spreadsheet format as defined in the Timing Diagram Editor Manual Section 11.11 Import from Spreadsheets. Basically the [Vectors] section is a tab separated format, where each column is a different variable. We have already created the file to save time, but you can view it.

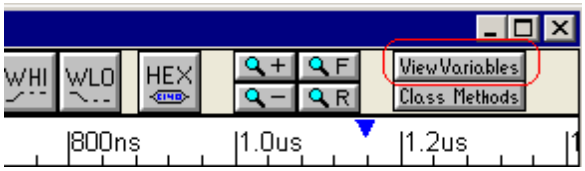
- Choose the **Report > Open Report Tab** menu to open a file dialog. Set **Files of Type** to **text (*.txt)**.
- Browse to the **SynaptiCAD\Examples\TutorialFiles\ReactiveTestBench \inputData** directory and select **inputData.txt** then press the **Open** button to open the file in the report tab.



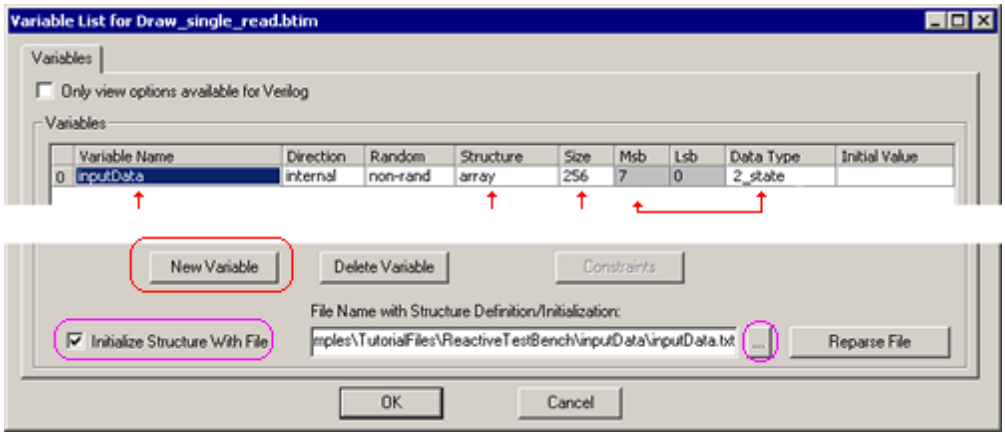
Copyright © 2014, SynaptiCAD Sales, Inc.

Create a Variable and Link it to a file:

- Click the **View Variables** button in the diagram to open the *Variable List* dialog.



- Press the **New Variable** button, then click on the name and change it to **inputData**. This name is important because it must match a column name in the input file that we choose.
- Under the *Structure* column double click and select **array**.
- Set *Size* to **256** to indicate that we are creating an array of 256 elements.
- Set *Data Type* to **2_state** then change *MSB* to **7** (this indicates we're storing 8-bit values in the array).



- Check the **Initialize Structure With File** checkbox near the bottom of the dialog.
- Browse to the **SynaptiCAD\Examples\TutorialFiles\ReactiveTestBench \inputData** directory and select **inputData.txt**.
- Press **OK** to close the dialog

Make the diagram reference the variable:

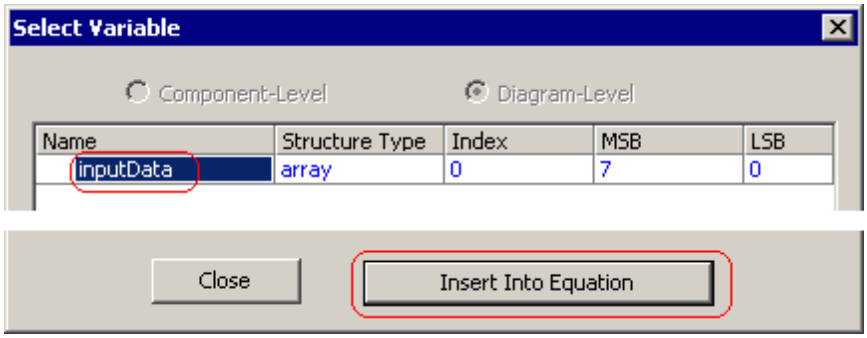
In the previous step we created an array variable called **inputData** that hooks up to a file that has a column titled **inputData**. Here we will make the DATA signal use the array to supply the signal values for both the write and the read cycles. In the next two sections we will add a variable called **address** which will be used to move through the array and a loop so that multiple write-read cycles can be performed without having to draw the transaction over and over again.

- Double-click on one of the AA states to open the *Edit Bus State* dialog then press the **Variable** button to open the *Select Variable* dialog.

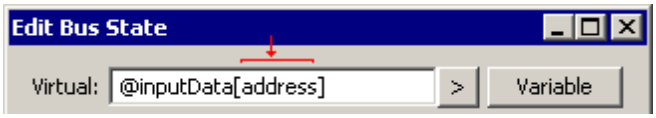


Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **inputData** variable, then press the **Insert Into Equation** button. This puts the variable into the ***Virtual*** box of the *Edit Bus State* dialog, so that you do not have to remember the exact syntax of the variable.

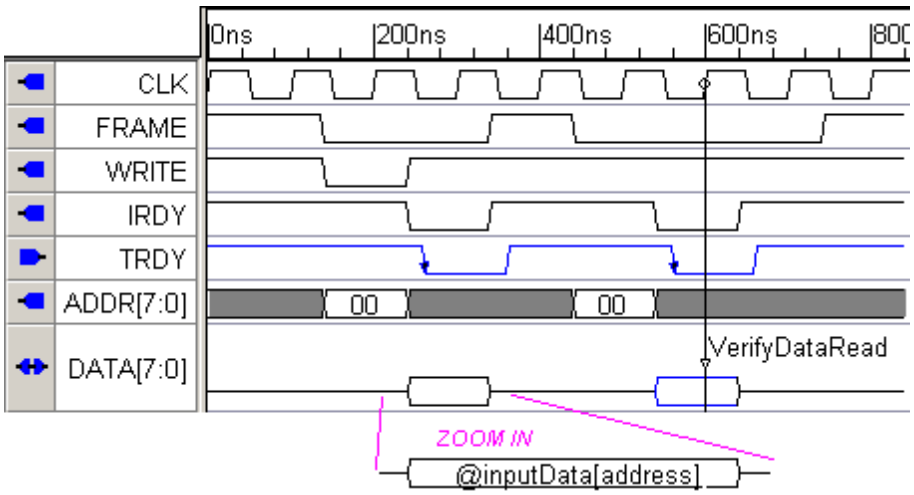


- Edit the equation so that it says**@inputData[address]**.The @ symbol is used to refer to a variable defined in the *Variable List* dialog. The **address** variable will be defined in the next section



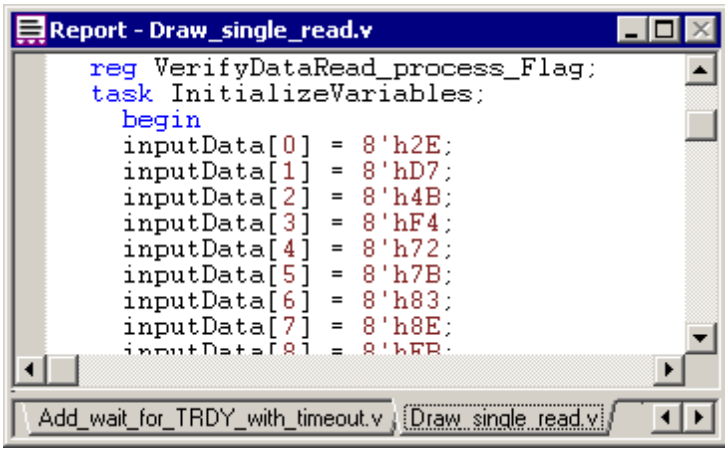
- Use the **next** or **previous** buttons to move to the other AA state and type in the same equation.

Below is a picture of the diagram. The variables do not show in the valid waveform at this zoom level because they are longer than the valid segments. If you zoom in you can see the variable names.



To see the generated code, export the test bench and view the code in the Report tab that contains the generated code. To see the code that initializes the inputData array, seach for "inputData". Next, look at the CLK_pos sequence, and search for "address" to find the assignment statement that drives the DATA bus with a value from inputData indexed by the address variable (the address variable will be created in the next step).

Copyright © 2014, SynaptiCAD Sales, Inc.



(TBench) 2.9 Create For-Loop to Perform Multiple Writes and Reads

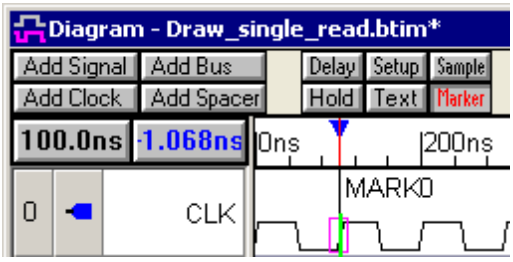
Loops and nested loops can be placed in the diagram to make the generated test bench apply the same test vectors. Here we will add a FOR loop that surrounds the write and read cycle and applys the waveforms 10 times.

Add the Loop Markers:

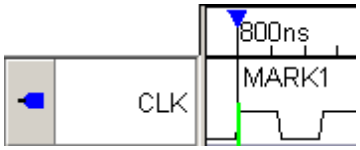
- Depress the **Marker** button so that right-clicks will add Marker lines to the diagram.



- **Left-click** the positive clock edge at **100ns** to select it, then **right-click** to place the Marker. This attaches the marker to the selected edge.



- Place another marker at the positive clock edge at **800ns** by first selecting the edge and then right clicking to add the marker.

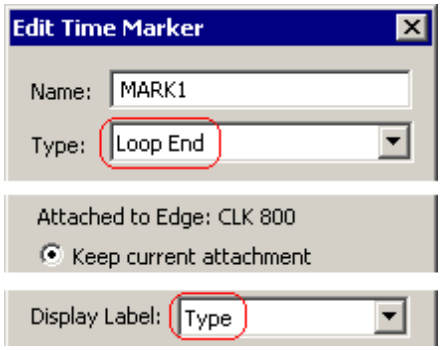


Copyright © 2014, SynaptiCAD Sales, Inc.

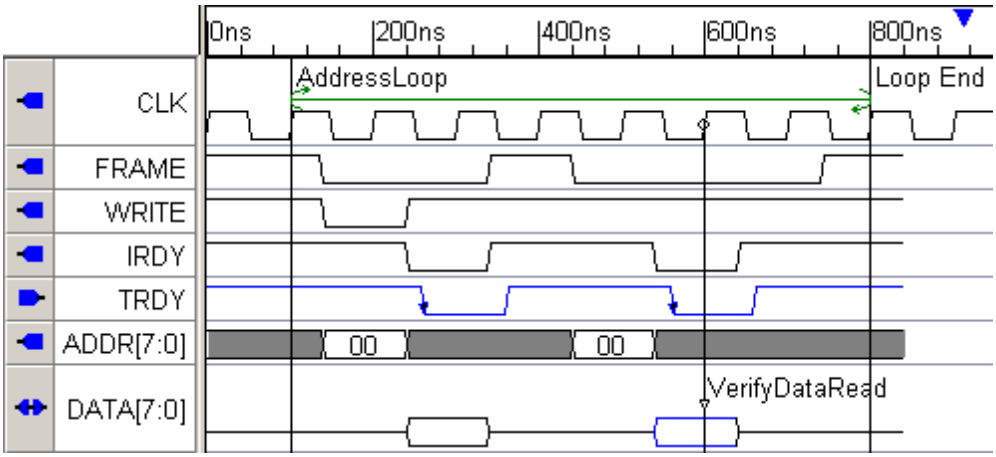
- Double-click on the first Marker to open the *Edit Time Marker* dialog.
- Change the name to **AddressLoop**, set the type to **For Loop**, set *Index* to **address,** and set end to **10**. This will define the variable address that we used in the previous section.
- Notice that the marker is attached to **CLK 100**. If you accidentally attached to the time you can use the controls in that section to attach to the clock.
- Then press **OK** to close the dialog.



- Double-click the second Marker to open the *Edit Time Marker* dialog.
- Set the *Type* to **Loop End**.
- Set the *Display Label* to **Type** so that the marker will display its type rather than its name.
- Notice that the marker is attached to **CLK 800**. If you accidentally attached to a time, you can use the controls in that section to attach to the clock.
- Press the **OK** button to close the dialog.

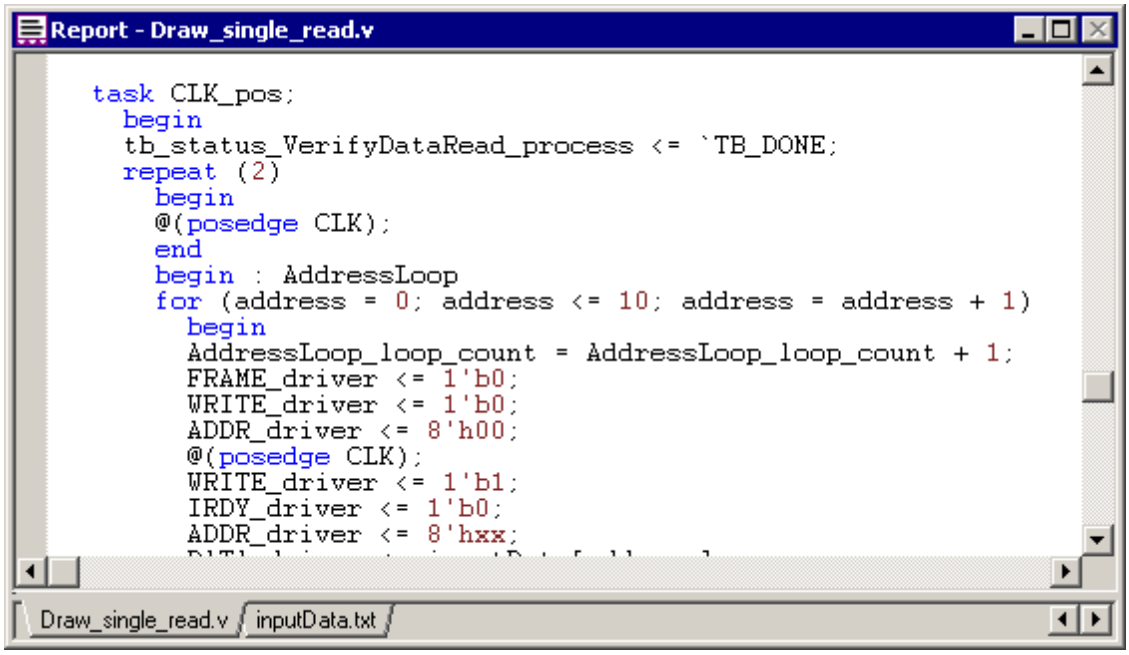


The two markers should now be connected graphically as shown below. During simulation the diagram between the loop markers will be applied ten times to the model under test. Each time a new value will be read from the file data and written to the model under test because of the **@inputData[address]**. Then the write will be verified during the read cycle. In this particular case we have written all the values to the same memory location but you could have also parameterized these values or read them in from the data file.



To see the generated code, export the test bench and view the code in the Report tab that contains the generated code. The for loop in the diagram generates a for loop in the generated code.

Copyright © 2014, SynaptiCAD Sales, Inc.

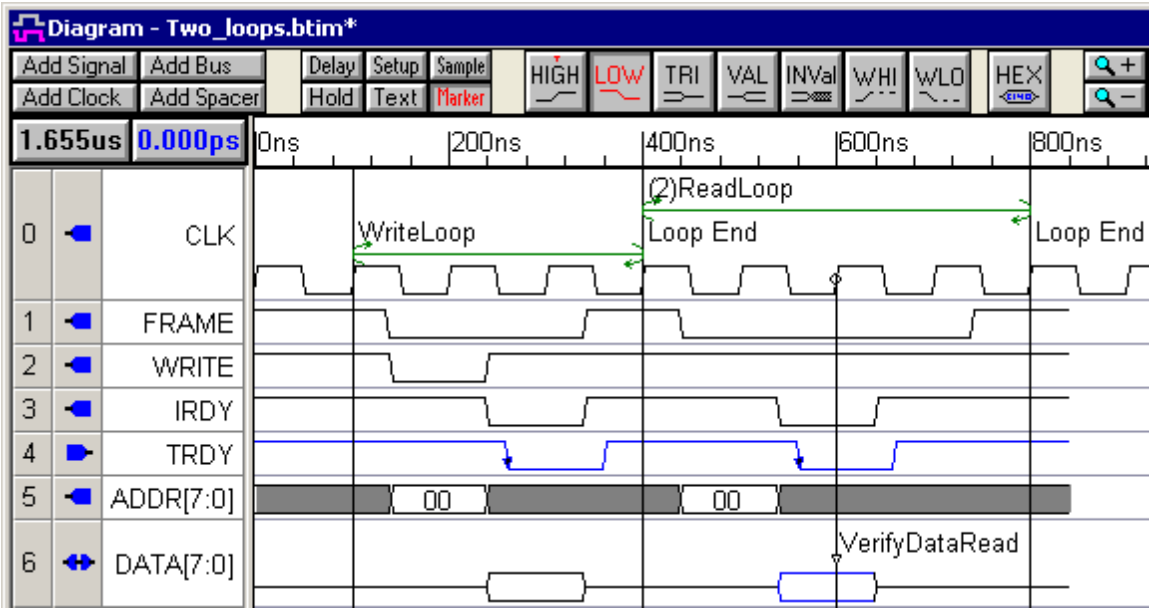


(TBench) 2.10 Alternate Test Bench Designs

The test bench for this tutorial has been a very simple one-loop read and write cycle. However with very small changes you can control the order of the reads and writes and also write random data instead of the data from the file. We will not perform these steps but you can look at the diagrams and see how it is done.

Alternalte: Consecutive Writes followed by Consecutive Reads

If you want to perform multiple writes concurrently, followed by multiple concurrent reads, then two for-loops are needed. The array of data can be referenced in each loop in the same manner already demonstrated. The (2) beside the ReadLoop label shows that it was added after the LoopEnd marker.



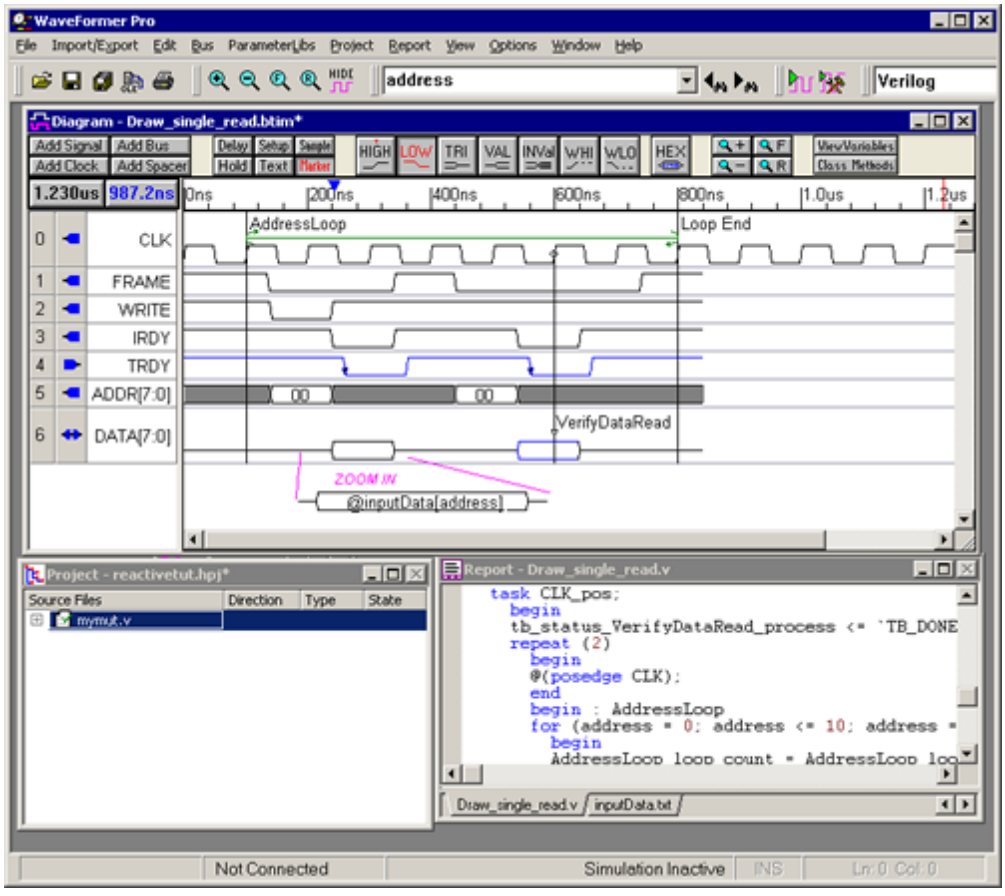
Copyright © 2014, SynaptiCAD Sales, Inc.

Random Data instead of the File data

In Verilog, you could use **$random()** as the state value for **DATA** during the write transaction. A user-defined function can also be embedded into the generated test bench using the *Class Methods* dialog which could be used to generate data values. In both of these cases, you would need to modify the state value under the **VerifyDataRead** sample since the **inputData** array is no longer used. A Sample must be placed on the driven **DATA** segment to capture the expected data. For example, you could create a Sample named **ExpectedData** that is triggered from the clock edge at **300ns**. Then the state under the **VerifyDataRead** Sample would be set to **ExpectedData** instead of **@inputData[address]**.

(TBench) 2.11 Summary of Reactive Test Bench Tutorial

Congratulations, you have completed the Reactive Test Bench Tutorial. You have created a self-testing test bench that uses a sample to test data coming back from the model under test. You have drawn both the stimulus vectors to drive the model under test and the expected waveforms that should come back from the model under test. You have experimented with edge sensitive control and with samples that block the test bench until a particular state is received. You have also added a loop that allows waveforms to be applied to the model under test without having to redraw the transactions over and over again.



Copyright © 2014, SynaptiCAD Sales, Inc.

Test Bench Generation 3: TestBencher Pro Basic Tutorial

In less then 30 minutes you will create a reusable test bench that can apply different stimulus and verify the results of a clocked SRAM. Below is a schematic of the different components that you will construct. First you will create the Project file that controls the generation of the interface model (test bench). Next you will draw the different transaction diagrams that are needed to communicate with the SRAM. And then you will edit the sequencer process to apply the transactions to the model under test. Finally you will simulate the design and verify the operation of the SRAM model.

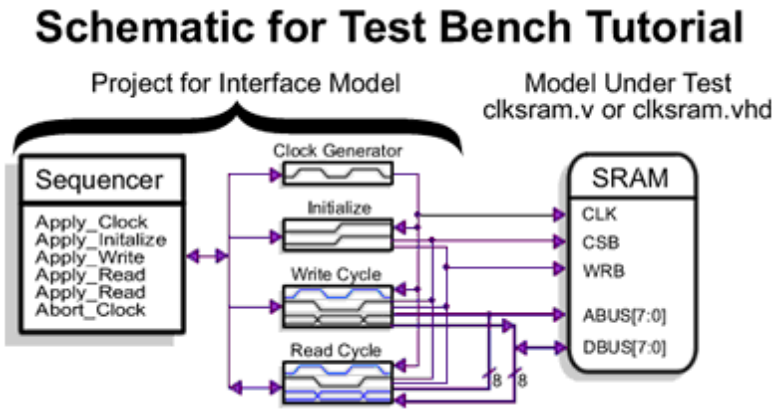


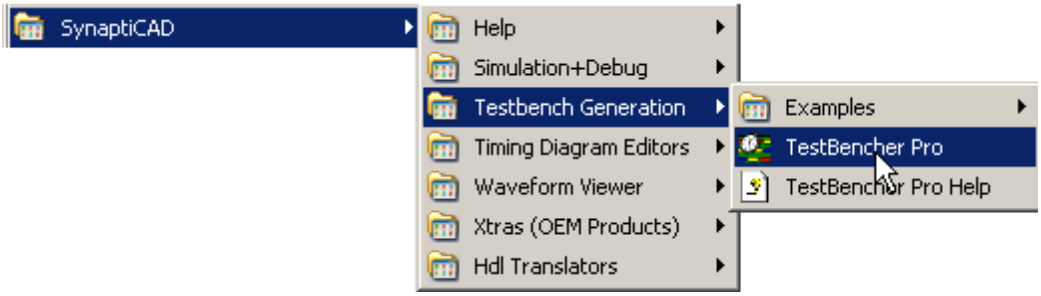
Figure 1: Tutorial Schematic

This tutorial assumes that you are familiar with the SynaptiCAD timing diagram editing environment. If you would like more information on the drawing environment then work through the short **Help > Tutorial > Basic Drawing and Timing Analysis** 10 tutorial.

(TBench) 3.1 Run TestBencher Pro

This tutorial requires a full version license for TestBencher Pro or an evaluation license. If you are evaluating then you can obtain a license by completing the form under the **Help > Request License** menu item and contacting our sales department.

- Run TestBencher Pro from the Start Menu.



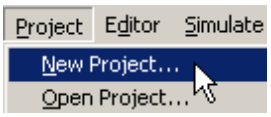
- To verify that you have a license, select the **Help > View License Details...** from the top-level menu of TestBencher to see if your license is detected.

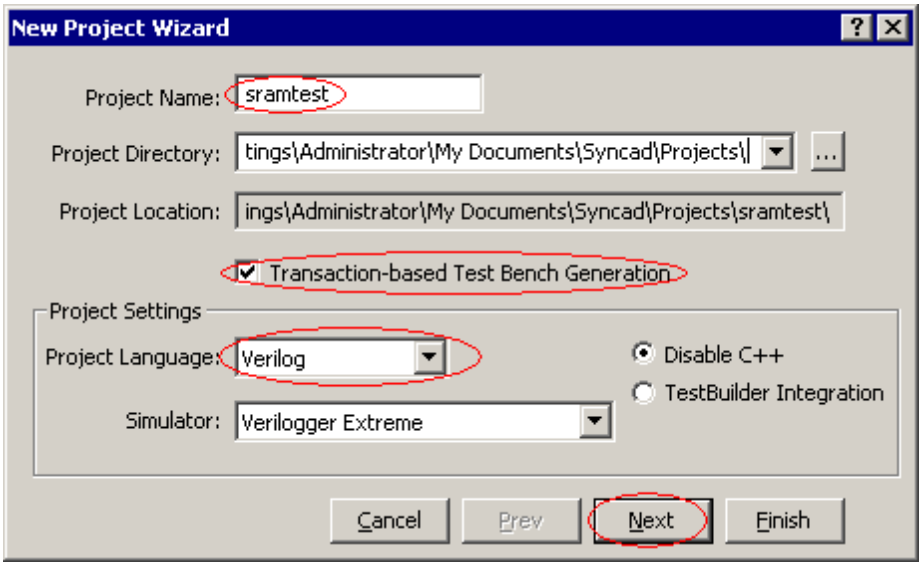
Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 3.2 Create a Project

TestBencher Pro uses a project file to represent and to control the generation of a bus-functional model (BFM) component. The information in the project file is displayed in the Project window and context sensitive menus (right-click menus) provide a list of actions that can be performed for the elements in the project tree. Projects are created using New Project Wizard dialog. This dialog helps setup the project directory, the generated language, and the clocking signal for the project.

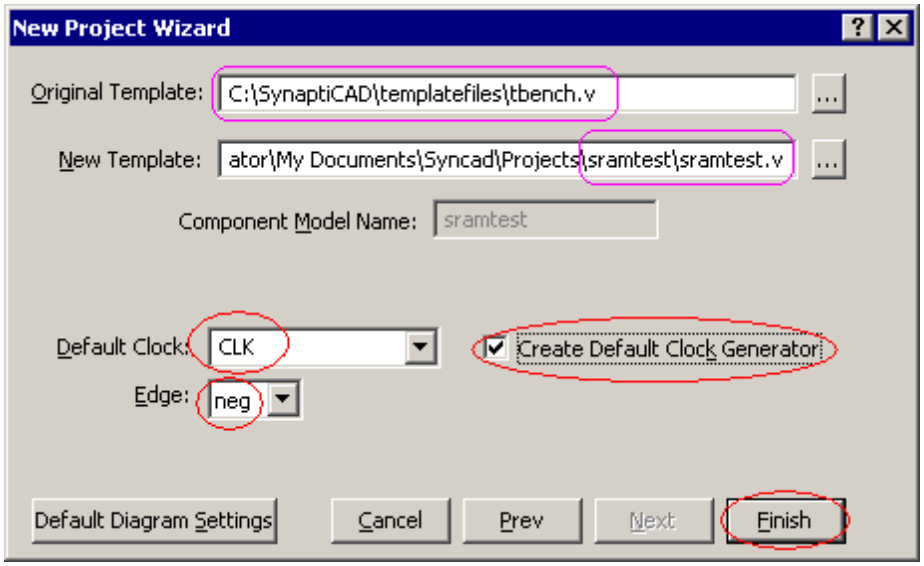
- Select the **Project > New Project** menu option to launch the *New Project Wizard* dialog.



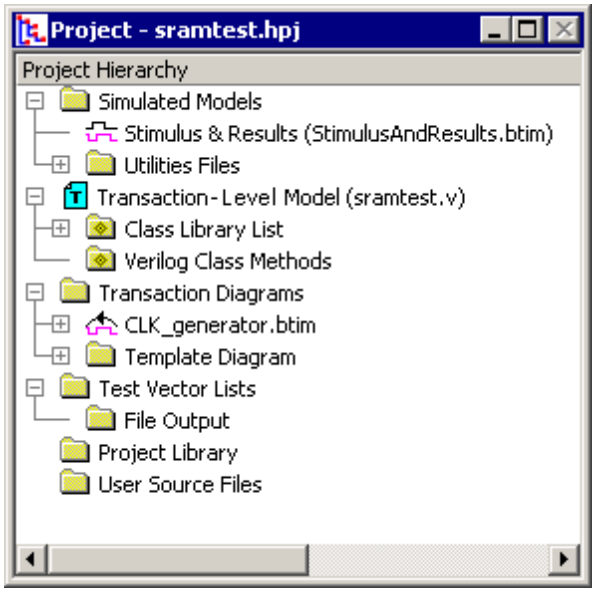


- Enter **sramtest** in the ***Project Name*** edit box. This will be both the name of the project and the subdirectory were project files are located. The subdirectory will be placed underneath the *Project Directory* path. **Unix users** need to make sure that you have read/write access to the directory specified in the *Project Directory* edit box.
- Check the **Transaction-based Test Bench Generation** checkbox.
- Use the ***Project Language*** box to select the code generation language. This tutorial can be use to generate Verilog, VHDL, and TestBuilder code. Sometimes a file name will be written as filename.<language extension>. This means that the file extension will be different depending on the language used: Verilog *.v, VHDL *.vhd, and TestBuilder *.cpp.
- Use the ***Simulator*** box to select your simulator. If you are evaluating, use the **Verilog** and **VeriLogger Extreme** combination, because the simulator is already set to work with TestBencher Pro. If you do not see your simulator pick one at random, then after the project is created read through the ***Chapter 1: Section 7: Setup External Simulators*** in the ***TestBencher Pro Manual*** to setup the simulator manually.
- Press the **Next** button to move to the second page of the *New Project Wizard*.

Copyright © 2014, SynaptiCAD Sales, Inc.



- Note that the name of the *New Template* is **sramtest** (the name of the project). TestBencher will use this file to generate the top-level module of the test bench. The *Original Template* is a file whose contents are copied into the new template file. Typically this file is **tbench.v** (a default file that ships with the software).
- Type **CLK** into the **Default Clock** box, and choose **neg** from the **Edge** box. Selecting a default clock causes the test bench to be cycle-based; if no clock is specified, the test bench will be event-based.
- Check the **Create Default Clock Generator** box. This will cause TestBencher to create a slave timing diagram called **Clk_generator.btim** that will drive the default clock signal.
- Press the **Finish** button to close the *New Project Wizard*, create the project, and populate the *Project* window.

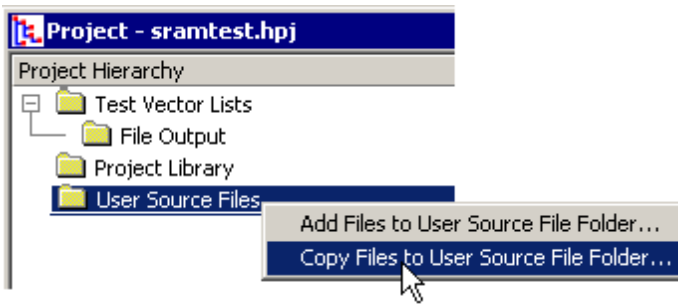


Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 3.3 Add the SRAM model to the Project

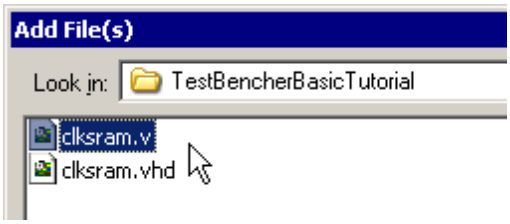
Next we will add the model under test (MUT) files to the project. TestBencher can parse the MUT files and extract the signal and port information for use in the transaction diagrams. Also, TestBencher uses the MUT information to instantiate the top-level component into the testbench model. For this tutorial we are going to test a clocked SRAM model.

- Right-click the *User Source Files* folder in the *Project* window and select **Copy Files to User Source File Folder** from the context menu option. This will open the *Add Files...* dialog.



- Use the **Look in** box to browse to the **SynaptiCAD > Examples > TutorialFiles > TestBencherBasicTutorial** directory.

- Depending on your language type, select either the Verilog **clksram.v** or the VHDL **clksram.vhd** file.



- Press the **Open** button to close the dialog and add the file to the project.



- Double click on the **clksram** MUT file to open the file in an editor window. Glance through the code so that you have an of how the model we will be testing works. Close the editor when you are done.

Copyright © 2014, SynaptiCAD Sales, Inc.

clksram.v

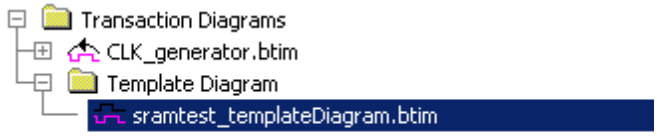
```
module clksram(CLK,CSB,WRB,ABUS,DBUS);

input CLK;
input CSB;
input WRB;
input [7:0] ABUS;
inout [7:0] DBUS;

reg  [7:0] DBUS_driver;
wire [7:0] DBUS = DBUS_driver;
reg [7:0] ram[0:255];

integer i;
initial                  //initialize all
   begin
   DBUS_driver = 8'bzzzzzzzz;
   for (i=0; i < 4095; i = i + 1)
     ram[i] = 0;
   end
```

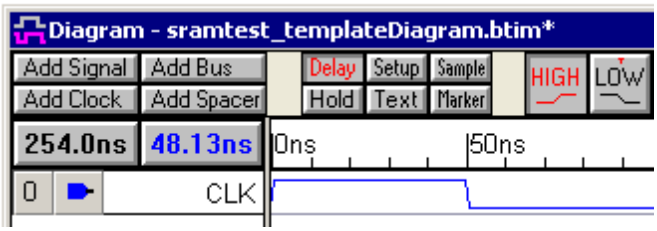
(TBench) 3.4 Setup the Template Diagram

When TestBencher created the project it also generated a template diagram. New transaction diagrams that are created for this project will contain the same signals, waveforms, parameters, and properties as the template diagram. Currently the CLK signal is the only signal in the template diagram. You are going to add the port signals for the clocked SRAM to the template so that later when you create the timing diagrams for the project all of the signal information matches up.

Extract the ports from the SRAM into the template diagram:

- In the *Project* window, under the *Template Diagram* folder, double click on **sramtest_templateDiagram.btim** to open the template diagram window.



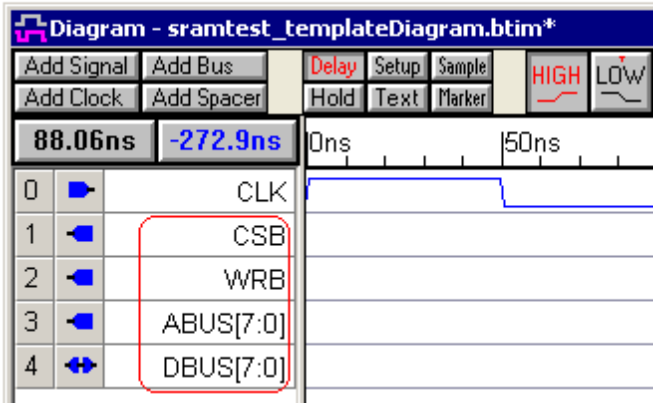


Copyright © 2014, SynaptiCAD Sales, Inc.

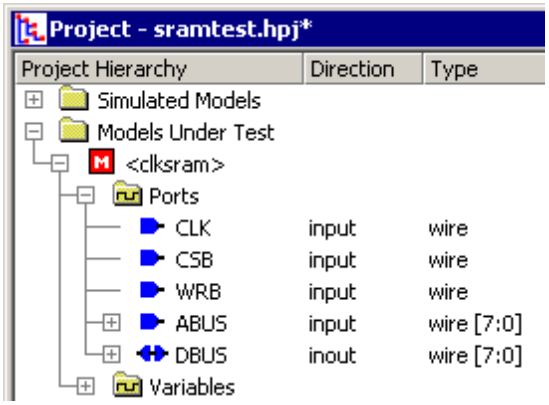
- Click the **Extract Ports from MUT** button. This will build the MUT and insert the signals for the MUT ports into the template diagram.



- The blue icons indicate the direction of the signals. DBUS has a direction of *inout*, because the data bus will need to both drive data to the SRAM model and receive data back from it. The CLK is an *input* to the diagram because it is driven by the CLK_generator diagram (the blue waveform also indicates it is an *input*). The rest are outputs of the test bench which will drive stimulus to the SRAM model under test.



- Notice that **<clksram>** is now present in the *Project* window under the *Models Under Test* folder. The single angle brackets indicate that clksram is the Model Under Test. Expanding this tree will display signal, port, and component information of the MUT.

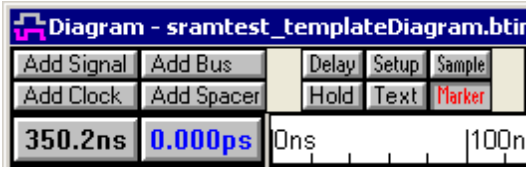


Note: If **<clksram>** was not generated as the MUT, then change the simulation preferences by choosing the **Options > Diagram Simulation Preferences** menu. Check the **Auto-create test bench and tree** check box. Press the **Extract Ports from MUT** button to rebuild the MUT.

Add an end diagram marker:

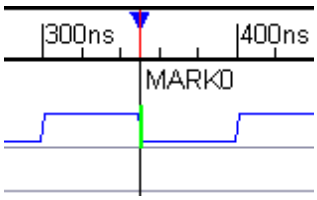
The transaction diagrams use an End Diagram Marker to indicate the exact time that the transaction ends. You can add an end diagram marker to the template diagram, so all new transactions will get the marker.

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



Copyright © 2014, SynaptiCAD Sales, Inc.

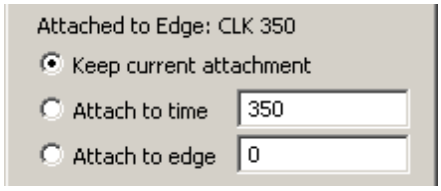
- Click on the fourth falling edge of the CLK signal (at 350ns) to select it and turn it green. Then right-click to add the marker line.



- Double-click on the marker to open the *Edit Time Marker* dialog.
- Select a Marker Type of **End Diagram** from the drop down list box. This end diagram marker will force the transaction to end at the fourth falling edge of the CLK signal.



- Notice that the Marker is **Attached to Edge** on **CLK** at 350ns. This is because you selected the edge before adding the marker. These controls can be used to changed the attachment.

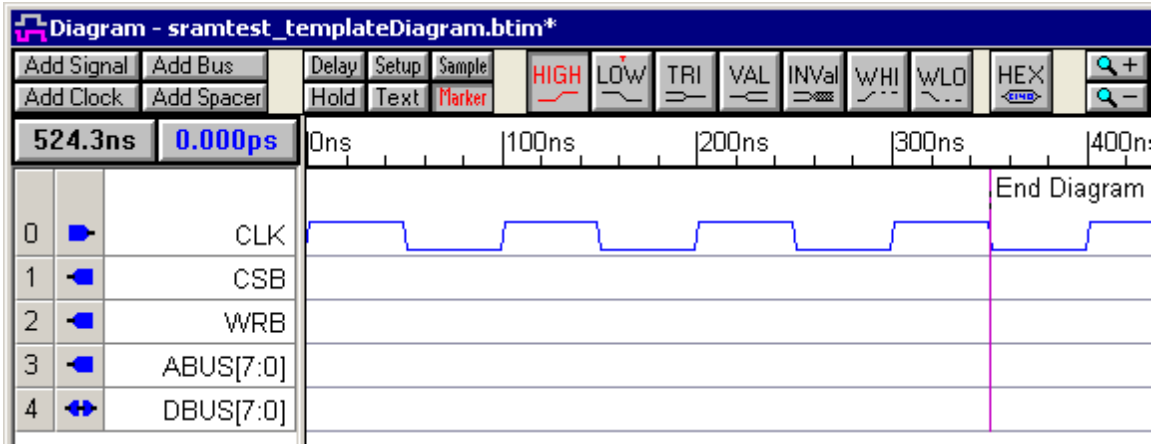


- Select **Type** from the *Display Label* list box. This will cause the marker to display its type rather then its name.



- Click **OK** to close the *Edit Time Marker* dialog.
- Use the **File > Save All Files** menu option to save the project and the template diagram.

The completed template diagram should look like the following:



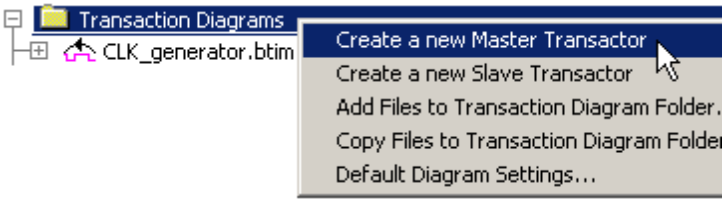
(TBench) 3.5 Create the Write Cycle Transaction Diagram

TestBencher Pro uses timing diagrams to represent reusable bus transactions. This tutorial will use two timing diagrams, **tbread.btim** and **tbwrite.btim**, to represent the read and write cycles used in testing the memory module. First, draw the write cycle diagram and then create variables for the data and address busses so that new values can be passed to the timing diagram each time it is called by the sequencer. Variables are also used to provide comparison values for runtime testing, and this will be demonstrated in the read diagram.

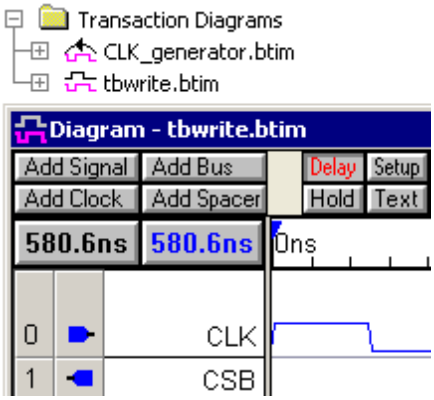
Copyright © 2014, SynaptiCAD Sales, Inc.

Create the Write diagram from the Template:

- In the *Project* window, right click the *Transaction Diagrams* folder and select **Create a new Master Transactor** from the context menu. This will cause the *Save As* dialog to open.

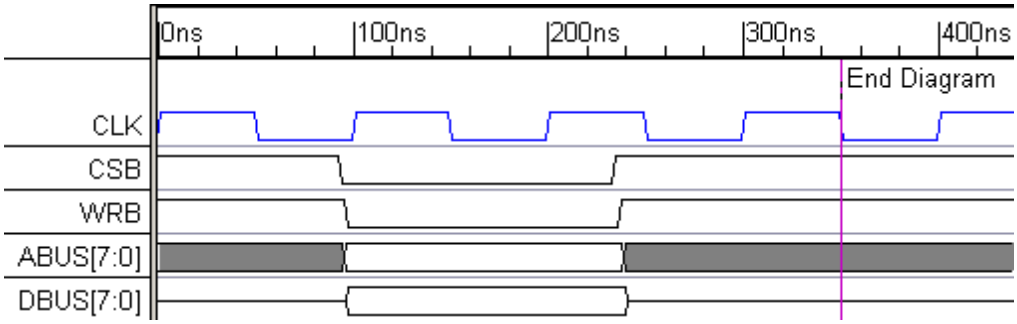


- Name the file **tbwrite** and press the **Save** button. This creates a new timing diagram using the information in the template file and lists the file in the *Transaction Diagram* folder.



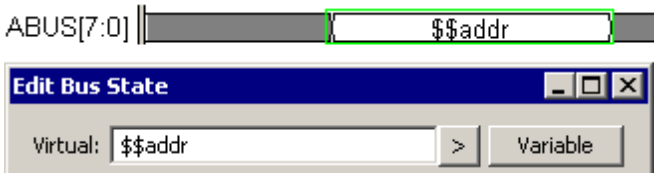
Draw the waveforms for the Write diagram:

Sketch the waveforms as shown on the diagram below. Since this is a negative edge clocked diagram the exact placement of signal edges is not important (unless it is near a negative clock edge). If you need a help drawing, refer to the Basic Drawing and Timing Analysis Tutorial sections **1.2 Drawing Signal Waveforms** 15.



Add the variables:

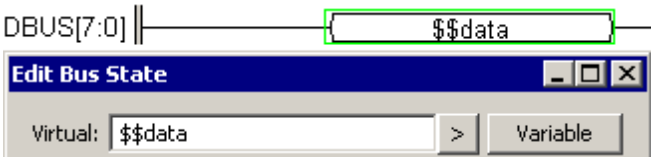
- Double click on the valid segment in the center of *ABUS* to open the *Edit Bus State* dialog.



- Type **$$addr** into the **Virtual** edit box. The "$$" in front of the variable name indicates that this is a state variable. If the "$$" is missing, TestBencher Pro will assume that this is the value of the address rather than a variable that will accept a value at a later time.

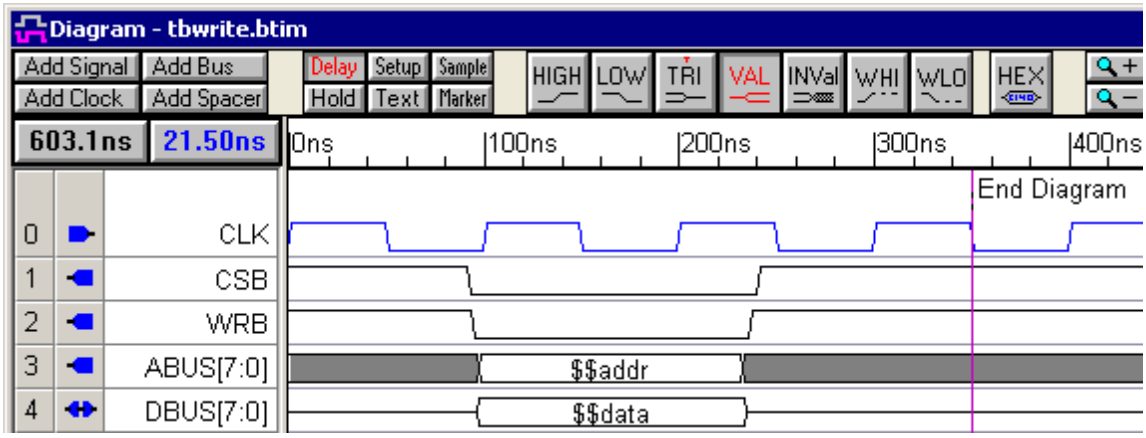
Copyright © 2014, SynaptiCAD Sales, Inc.

- Click on the valid segment in the center of *DBUS* to move the focus of the *Edit Bus State* dialog to the new segment.



- Type **$$data** in the **Virtual** edit box, then press the **OK** button to close the *Edit Bus State* dialog. The two edited segments will display the state variables.
- Click the diskette icon on the main toolbar to save the timing diagram.

Below is the completed write transaction. When the chip select (CSB) and the write enable (WRB) are low, the address and data busses will be driven with the current values of the $$addr and $$data variables.



(TBench) 3.6 Create the Read Cycle Transaction Diagram

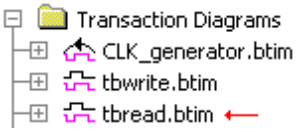
The read cycle will initiate a read transaction with the clocked SRAM and also monitor the data bus to verify the result of the read. Since the signals for the read diagram are so similar to the write diagram, we will start by copying the write diagram and then modify the waveforms (instead of starting with the template).

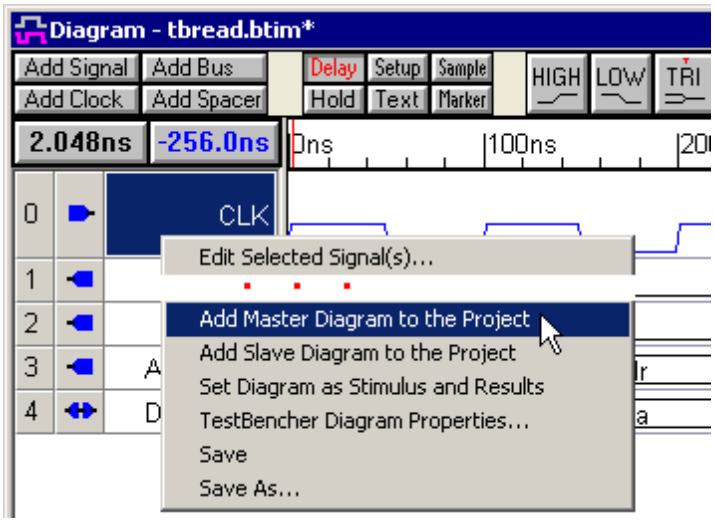
Copy the tbwrite diagram to make tbread:

- Click on the ***tbwrite diagram*** window bar so that it is the active window, then choose **File > Save Timing Diagram As** menu to open the *Save As* dialog
- Name the file **tbread**, and press the **Save** button.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Right-click in tbread's Label window, and select **Add Master Diagram to Project** from the context menu. This will add tbread to the *Transaction Diagrams* folder in the *Project* window.





Modify the WRB Signal:

The write control signal, WRB, should stay high (inactive) for the duration of the read.

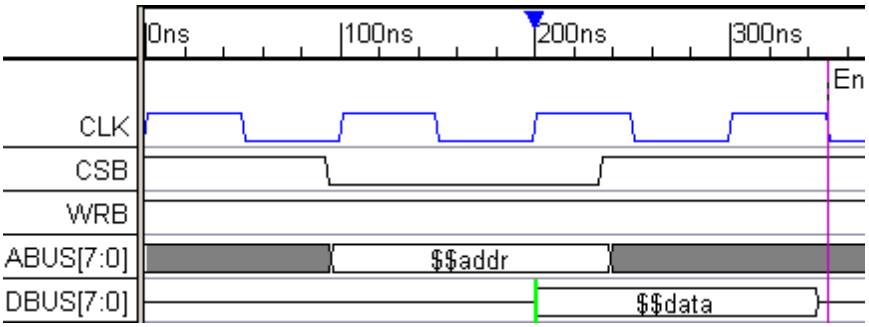
- Select the center segment and press the delete key to remove the low signal segment.



Modify the DBUS Signal:

Since our SRAM is clocked the data comes out on the clock cycle after the chip select signal, CSB, goes active. You can drag each edge of DBUS individually, or use the following technique to shift the whole signal.

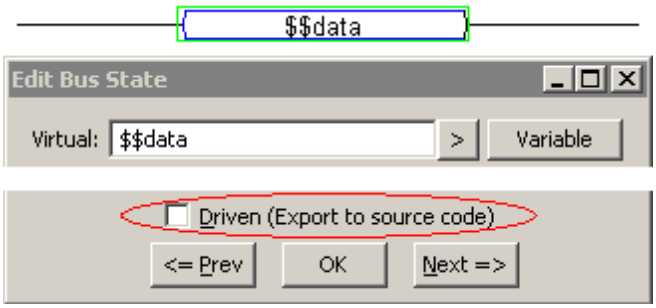
- Shift the start of the DBUS data segment to 200ns. Hold down the <2> key (the number 2 key) on the keyboard, while dragging the starting transition to 200ns. The <2> key causes transitions to the right of the selected edge to move with the dragged edge.



During the read transaction, the DBUS signal will be driven by the SRAM so it will be an input signal to the test bench (not an output like in the write cycle).

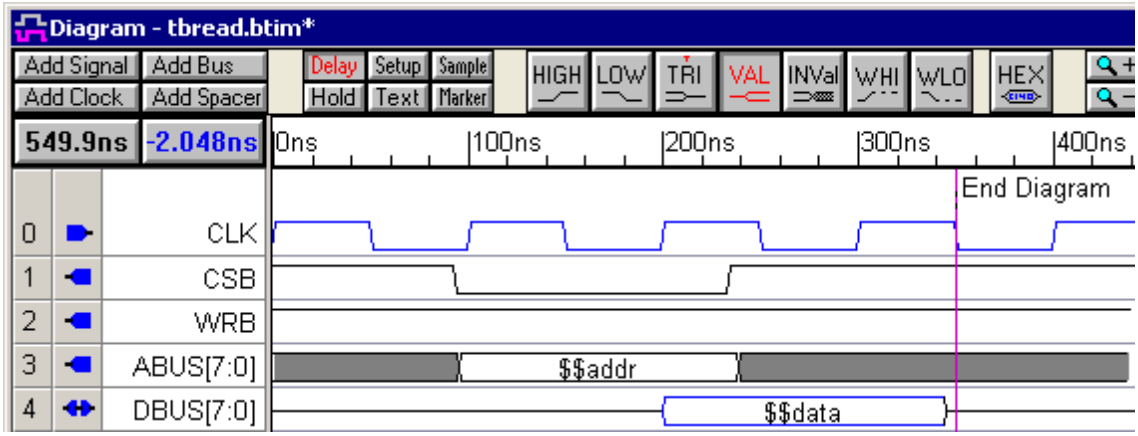
Copyright © 2014, SynaptiCAD Sales, Inc.

- Double click on the data segment to open the *Edit Bus State* dialog.
- Uncheck **Driven (Export to source code)** checkbox.This will cause the segment to be displayed in blue.



- Click the diskette icon on the main toolbar to save the timing diagram.

The completed read diagram looks like the following:



(TBench) 3.7 Add a Sample to Verify Data

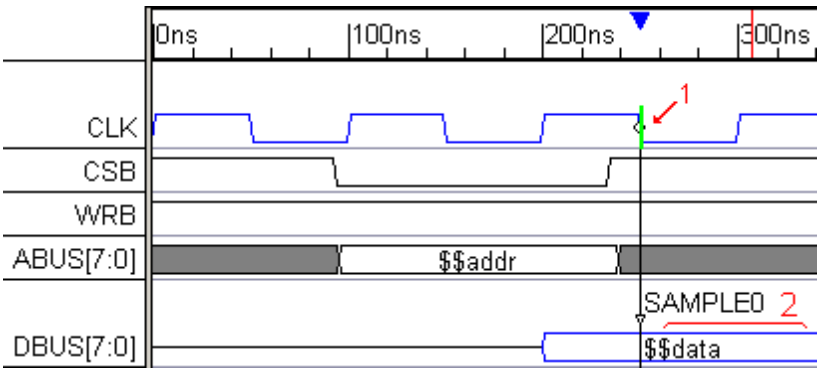
Next a Sample will be added to the read timing diagram. Samples compare the actual state value of an input signal to the expected state value, and conditionally react to the results of the comparison.

Add the sample:

- In the **tbread** diagram, press the **Sample** button so that right clicks will add samples.



- First, left-click on the **third falling edge** (250ns) of *CLK* to select the edge.
- Then, Right-click near the end of the **blue valid segment** on *DBUS*.



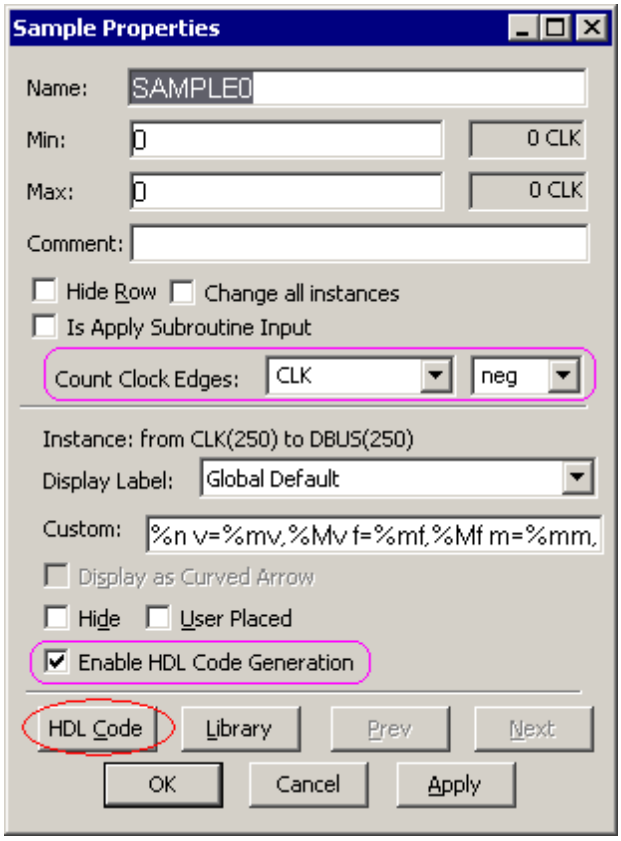
- This adds a Sample parameter named **SAMPLE0** that lines up with the third neg edge of the *CLK* signal.

Copyright © 2014, SynaptiCAD Sales, Inc.

Investigate the sample code generation features:

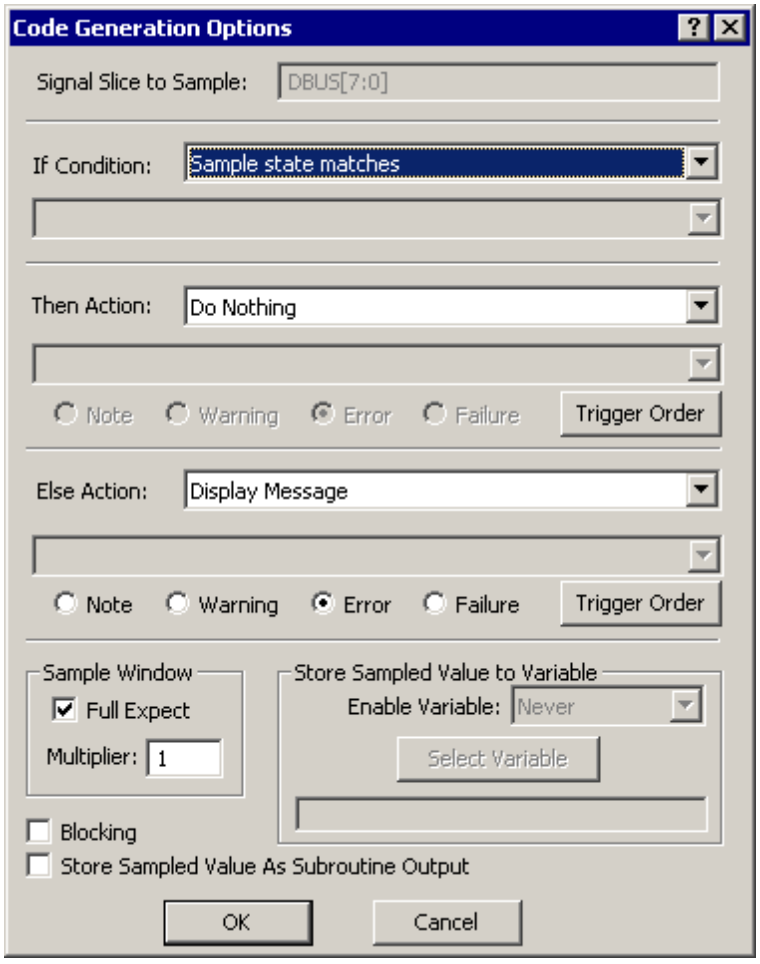
The default behavior of the sample compares the run time value with the drawn value ($$data) and throws an *Error* if they are different. This is the behavior that we need for the tutorial. The next few steps show you the HDL code generation dialog and how to control the generated code. You do not need to make any changes to the dialog defaults.

- Double-click on the sample name **SAMPLE0** in the drawing window to open the *Sample Properties* dialog.
- Notice that this dialog controls all of the display features for the Sample. The sample can be offset from the triggering edge. Also notice that the HDL Code Generation is enabled.
- Press the **HDL Code** button to open the *Code Generation Options* dialog.



Copyright © 2014, SynaptiCAD Sales, Inc.

- The Sample generates an code in the form of if/then/else.
- Look through the *condition* and *action* drop-downs to see the built-in behavior choices.
- Make sure to return the dialog to the shown state so that error messages will be generated if the Sampled data does not match the variable.



- Press **OK** to close the *Code Generation Options* dialog. Then press **OK** to close the *Sample Properties* dialog.
- Save the timing diagram by selecting **File > Save Timing Diagram** from the main TestBencher menu.

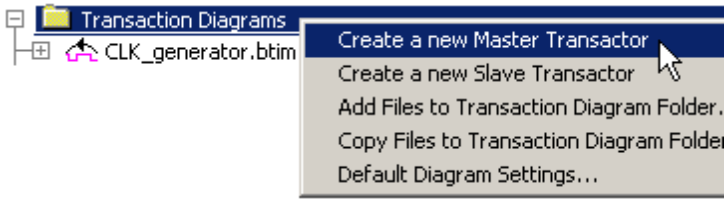
(TBench) 3.8 Create the Initialize Transaction Diagram

When drawing the waveforms for a transaction diagram it is important to remember that transactions do not automatically include an event at time zero and that only the drawn events are driven. This is a feature that allows transactions to be reused any time during simulation without implying any initialization information. In our example, the clocked SRAM control signals, CSB and WRB, need to be initialized before the read and write cycles are applied to the model. We will draw a simple initialization diagram that will drive the control signals to high (inactive).

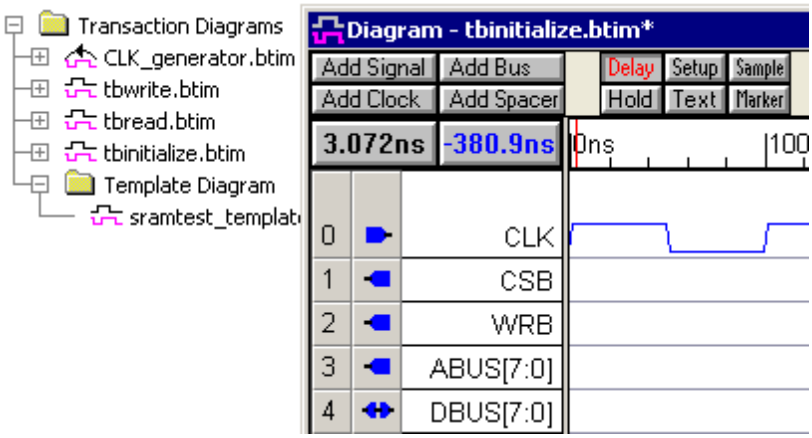
Copyright © 2014, SynaptiCAD Sales, Inc.

Create the Initialization diagram from the Template diagram:

- In the *Project* window, right click the *Transaction Diagrams* folder and select **Create a new Master Transactor** from the context menu. This will cause the *Save As* dialog to open.



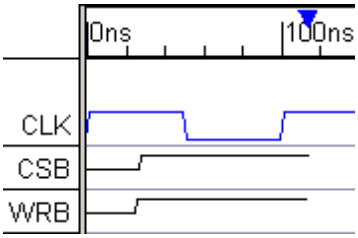
- Name the file **tbinitialize** and press the **Save** button. This creates a new timing diagram using the information in the template file and lists the file in the *Transaction Diagrams* folder.



Edit the Waveforms:

- Remove the **ABUS** and **DBUS** signals, because the tri-state bus signals do not need to be initialized. Select the **ABUS** and **DBUS** signals by clicking on them, and then press the **<delete>** key to delete the selected signals.

- Draw the following waveforms as shown (tristate for 10ns then high). If you need a help drawing, refer to the Basic Drawing and Timing Analysis Tutorial sections **1.5 Drawing Signal Waveforms** 15.

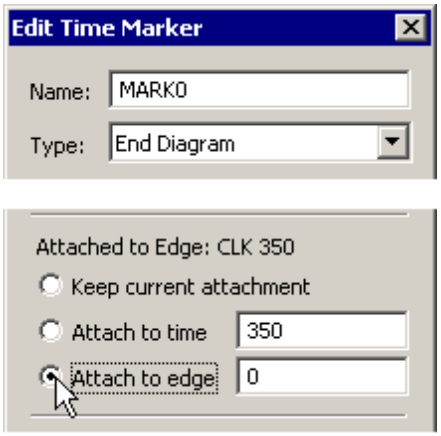


Move the End Diagram Marker:

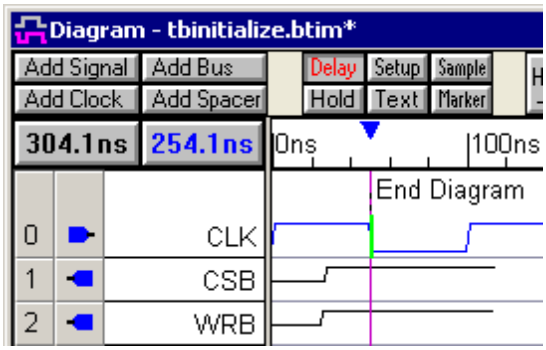
The initialization timing diagram will only need one clock cycle to initialize the control signals. Therefore, the **End Diagram** marker can be moved to the 1st negative clock edge.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Double-click on the marker to open the *Edit Time Marker* dialog.
- Select **Attach to Edge** from the radio buttons.
- Click **OK** to close the *Edit Time Marker* dialog. This will put TestBencher into a special select mode.



- As you move the cursor around in the diagram a green bar will jump to the closest edge to remind you that you are in the Attach to edge mode. Click on the first negative clock edge (at 50ns) to attach the marker to that edge.



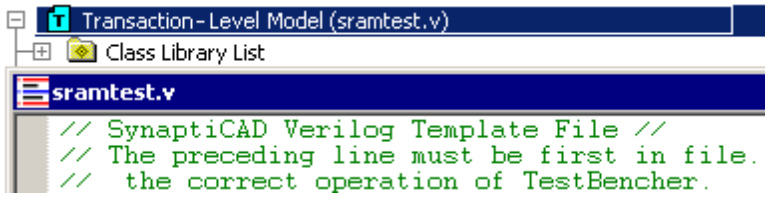
- Click the diskette icon on the main toolbar to save the timing diagram.

(TBench) 3.9 Add Transaction Calls to the Sequencer Process

Inside the primary template file for the project is a *Sequencer Process*. This process is the place in the top-level test bench that defines the order in which the timing transactions are applied to the model under test. By using the *Insert Diagram Calls* dialog, you can construct the testbench by writing little to no code.

Open the Transaction-Level Model for the main Test Bench:

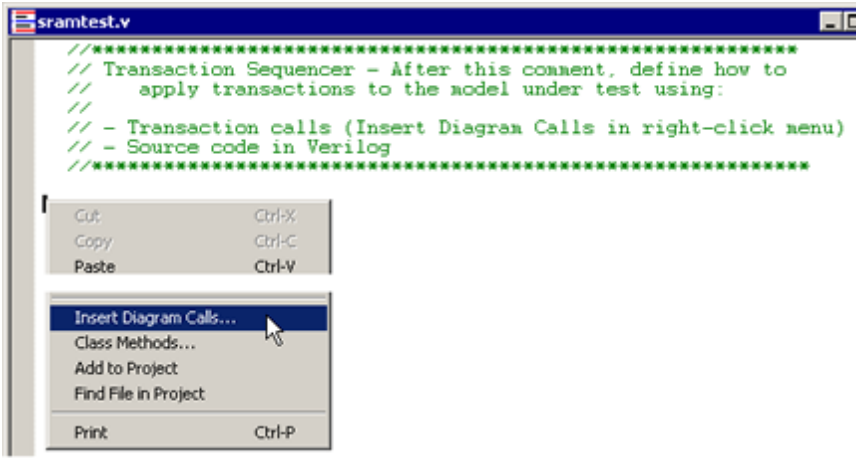
- In the *Project* window, double click on the *Transaction-Level Model* folder to open an editor window with the **sramtest** template file.



Find the Transaction Sequencer in the Component Model:

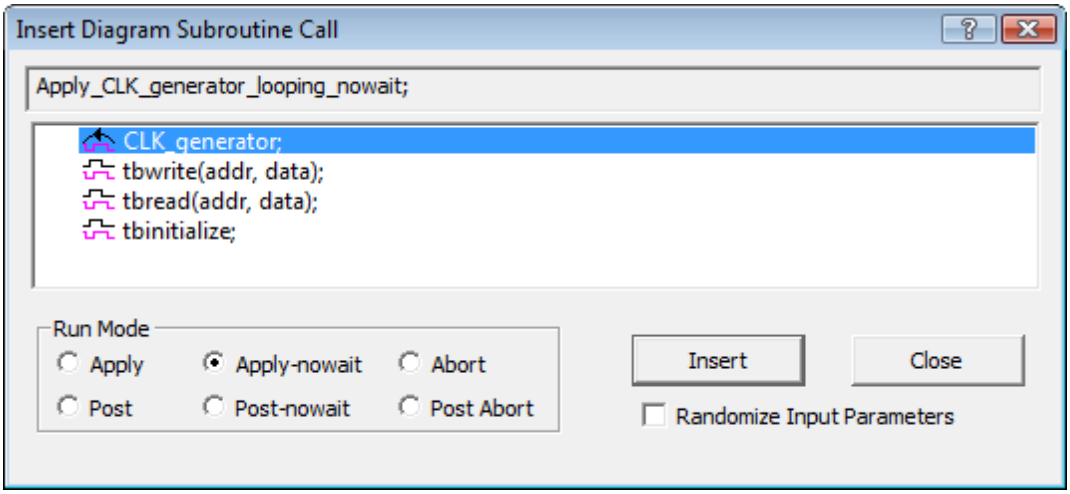
- Scroll down in the **sramtest** editor window near the end of the file until you find the **Transaction Sequencer** comment block. The comment changes depending on the code generation language.

Copyright © 2014, SynaptiCAD Sales, Inc.



Open the Insert Diagram Calls Dialog:

- Click in the **sramtest** editor window below the Transaction Sequencer comment (as shown above) so that the blinking cursor is in the place where the apply statement should be added.
- Right-click in the editor window and select **Insert Diagram Calls** to open the *Insert Diagram Subroutine Call* dialog.



- Arrange the windows so you can see the editor and the dialog at the same time.

Insert the Apply calls:

The *Insert Diagram Subroutine Calls* dialog generates diagram apply calls so you do not need to memorize the function syntax. Each timing diagram can generate several task calls:

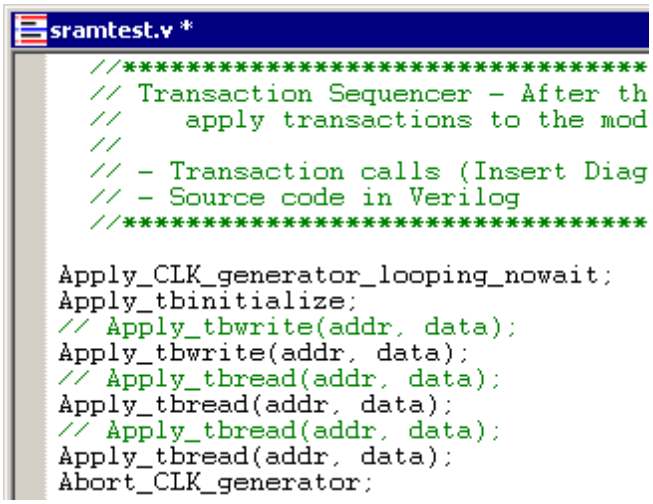
- **Apply** runs the transaction in a blocking mode so that no other transactions will run until this one is done.
- **Apply-nowait** runs the transaction concurrently with other transactions.
- **Abort** stops a running transaction.
- The **Post** calls are supported only in VHDL. A Post call schedules a transaction for future execution by placing it into a Transaction Manager queue (instead of immediately applying the transaction). The Transaction Manger must be enabled as described in Section 7.3

Copyright © 2014, SynaptiCAD Sales, Inc.

Transaction Manager Overview for these calls to operate. We will not be using the Post calls in this tutorial.

The Master/Slave *Diagram Setting* determines how many times a transaction executes. Master Transactors, like the Read, Write, and Initialize diagrams run once and stop. Slave Transactors like the Global Clock Generator run in a looping mode until an Abort call is received. You will first start the clock, initialize the control signals, write to the SRAM, read from the SRAM twice, and then abort the clock.

- Double click on the **CLK_generator** insert the statement into the Sequencer Process. Since this is a slave diagram (indicated by the black arrow), the default state is **Apply-nowait**, because most of the time slave diagrams will run concurrently with other diagrams.
- Double click on the **tbinitialize** entry. Since this is a master diagram, the default state is **Apply**, because usually Master diagram run in blocking mode.
- Double click on the **tbwrite** entry.
- Double click on the **tbread** entry TWO times to insert the code to add two read calls.
- Select **CLK_generator** entry, choose **Abort** radio button, and then press the **Insert** button to insert the code. This will add the abort call to stop the clock generator. Without this call, YOUR SIMULATION WOULD CONTINUE TO RUN FOREVER since the clock wouldn't stop.
- Close the *Insert Diagram Subroutine Call* dialog.
- The apply calls should look similar to the following code block. Different languages may have extra parameters.



Edit the State Values of the Write and Read Apply calls:

Edit the write and read Apply code lines and replace the state variable names with actual variables that will be passed into the timing diagrams. The comment lines are there to document the parameter variable names. **Note:** The code to be entered is **bold**.

- For Verilog type:

```
Apply_tbwrite('hF0, 'hAE);
Apply_tbread('hF0, 'hAE);
Apply_tbread('hF0, 'hEE);
```

- For VHDL type:

Copyright © 2014, SynaptiCAD Sales, Inc.

```
Apply_tbwrite(tb_Control, tb_LocalBfmPath, x"F0", x"AE");
Apply_tbread(tb_Control, tb_LocalBfmPath, x"F0", x"AE");
Apply_tbread(tb_Control, tb_LocalBfmPath, x"F0", x"EE");
```

- For OpenVera type:

```
tb_tbwrite.ExecuteOnce('hF0, 'hAE);
tb_tbread.ExecuteOnce('hF0, 'hAE);
tb_tbread.ExecuteOnce('hF0, 'hEE);
```

Notice that the **tbwrite** apply statement writes the hex value AE to memory cell F0. The **tbread** diagram calls will then read the value from the same memory cell. The data values provided in the **tbread** diagram calls will be used to compare with the actual value. The first call to **tbread** will expect to find a value of hex AE in the address F0. The second call to **tbread** will expect to find the hex value EE instead. This will cause the sample to report an error during the second execution of **tbread**.

- Save the top-level template file by right-clicking in the editor window and selecting **Save**.

In addition to these task calls, you can also place HDL code in the sequencer. One example where this would be useful is if you wish to place conditions on whether or not a timing transaction is called, or on the parameter values that you wish to have applied.

An alternative method to placing transaction calls in the sequencer process is to create a file external to the bus-functional model with transaction calls and during simulation read the transaction calls from a file (see Section 7.3: Transaction Manager Overview in the online TestBencher Manual).

(TBench) 3.10 Setup the Simulator

At this point all the timing diagrams have been created and you have edited the Sequencer process. Next we will generate the test bench and simulate the entire design, but we should first check to see if the simulator is setup properly.

TestBencher can control external simulators and compilers or use its built-in Verilog simulator to compile and simulate the design. If you are using the built-in simulator, skip ahead to next section. *Step 7: Setup External Simulators* in the TestBencher Pro online manual has a complete list of instructions for working with remote simulators and for setting up a compiler for TestBuilder.

To configure a third-party simulator:

- Choose the **Options > Simulator and Compiler Settings** menu option. This will open the *Simulator and Compiler Settings* dialog.
- From the **Simulator and Compiler tools** drop-down select the appropriate simulator.
- Enter the directory that contains the simulator executable in the **Simulator Path** edit box.
- Press **Compile Syncad Libraries** to build libraries required by the simulator in order to compile TestBencher projects. **IMPORTANT**: If you omit this step, you will get compile errors when you attempt to compile your test bench source files. This has to be done ONCE for each new simulator.
- Click **OK** to close the *Simulator and Compiler Settings* dialog.

Select the third-party simulator:

- Select the **Project > Project Simulation Properties** menu option. This will open the *Project Simulation Properties* dialog.
- Select the tab for the language you are working with.

Copyright © 2014, SynaptiCAD Sales, Inc.

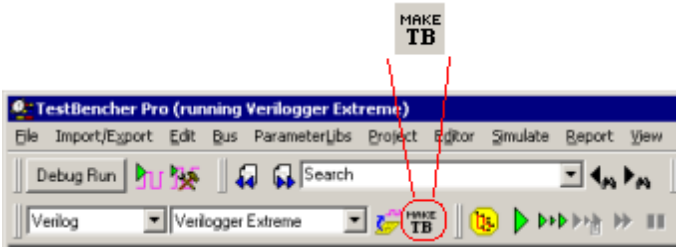
- Select the desired simulator from the **Simulator Type** drop-down. If you are working in VHDL and using ModelSim XE/PE (but not SE), you should probably set this value to ModelSIM VHDL GUI, because XE and PE do not support the API required for TestBencher to capture the waveforms directly inside the GUI (the API is supported by ModelSim SE).
- Click **OK** to close the *Project Simulation Properties* dialog.

(TBench) 3.11 Generate the Test Bench and Simulate

Once the simulator is setup you are ready to generate the test bench and simulate the design.

To generate the test bench:

- Press the **Make TB** button. This will expand the macros in the template file and pop up a dialog that says "*Finished generating test bench. Please check waveperl.log for errors*." Close this dialog by clicking the **OK** button.



- In the *Report* window, check the **waveperl.log** tab to see if TestBencher encountered any errors during the test bench generation. If it did, fix the error and regenerate the test bench. (If you can not see the *Report* window, choose the **Window > Report** menu to bring it to the front.)

To simulate the design:

- Click the yellow **Compile Model and Test Bench** button. This builds (parses) the project using the tools specified in the *Project Settings* and *Simulator and Compiler Settings* dialogs.



- In the bottom right corner, a yellow **Simulation Built** status message indicates the build was successful and that you are ready to simulate. If the status indicates an error, the *Report* window *Errors* tab displays the compile errors. If there are errors then fix them, regenerate the test bench, and recompile.



- Click the green run button on the simulation button bar. This will simulate the design and display the results in the *StimulusAndResults* diagram and the *Report* window *simulation.log* tab.



- In the bottom right corner, a **Simulation Good** status message indicates that the simulation has reached a successful end.

(TBench) 3.12 Examine Report Window Results

The *Report* window **simulation.log** tab displays the default log file for the simulator. TestBencher automatically writes a message to the log file each time a transaction starts and stops. The clocked SRAM contains code to display a message each time it performs a read or write. We also added a

Copyright © 2014, SynaptiCAD Sales, Inc.

sample parameter to the Read Cycle, and set it to generate an error message when the data from the SRAM does not match the expected value.

Examine the log file:

- In the *Report* window, open the **simulation.log** tab and display the following results:

```
sim> run
SIM: TB> Note: In "sramtest_CLK_generator" at 0.000ns: Executing LOOPING
SIM: TB> Note: In "sramtest_tbinitialize" at 0.000ns: Executing ONCE
SIM: TB> Note: In "sramtest_tbinitialize" at 50.000ns: Execution DONE
SIM: TB> Note: In "sramtest_tbwrite" at 50.000ns: Executing ONCE
SIM: In clksram at 150.000ns: Writing ae to address f0
SIM: TB> Note: In "sramtest_tbwrite" at 350.000ns: Execution DONE
SIM: TB> Note: In "sramtest_tbread" at 350.000ns: Executing ONCE
SIM: In clksram at 450.000ns: Reading ae to address f0
SIM: TB> Note: In "sramtest_tbread" at 650.000ns: Execution DONE
SIM: TB> Note: In "sramtest_tbread" at 650.000ns: Executing ONCE
SIM: In clksram at 750.000ns: Reading ae to address f0
SIM: TB> Error: In "sramtest_tbread" at 850.000ns: Sample SAMPLE0_process
sampled signal: DBUS
SIM:       Expected state: ee ; Detected state: ae
SIM: TB> Note: In "sramtest_tbread" at 950.000ns: Execution DONE
SIM: TB> Note: In "sramtest_CLK_generator" at 950.000ns: Execution DONE
SIM: TB> Total Warnings =           0
SIM: TB> Total Errors =           1
Simulation finished due to event queue exhaustion.
Simulation time: 0.2 s (CPU time: 0.2 s)
sim> start_corba_msg_pump
sim> exit
Process exited with code 0.
```

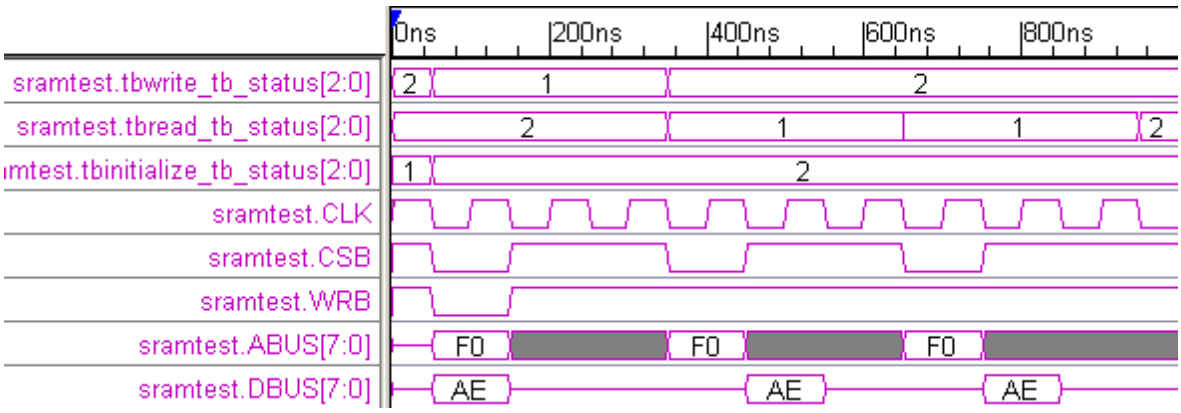
- Notice that the clock generator starts at time zero and continues until the end of the simulation when the abort call is issued.
- The initialization diagram also starts executing at time zero and blocks the next transaction until it is complete.
- The write diagram starts next and writes a value to the SRAM. The SRAM acknowledges that is writing the value to the specified address.
- The first read diagram executes successfully.
- The second read diagram throws a warning because the expected value did not match the value from the MUT. We purposely passed in a bad expected data value so we could see how the sample throws the error.
- Next the abort call to the clock stops the clock transaction and ends the simulation.

Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 3.13 Examine the Stimulus and Results Diagram

After simulation the Stimulus and Results diagram will contain all of the top level signals of the project, the driver signals, and status and trigger signals for each transaction.

- Hide some of the signals in the Stimulus and Results diagram by selecting the signal names and choosing **View > Hide Selected Signals** until the diagram looks like this:



- A status signal of <1> indicates the transaction is running. You can see that the initialization diagram runs followed by the write cycle and two read cycles.
- During the write cycle, the data AE is written to address F0. When comparing the simulated write cycle to the drawn transaction, remember that this is a negative clock edge diagram.
- The read cycles read back the data from the memory.

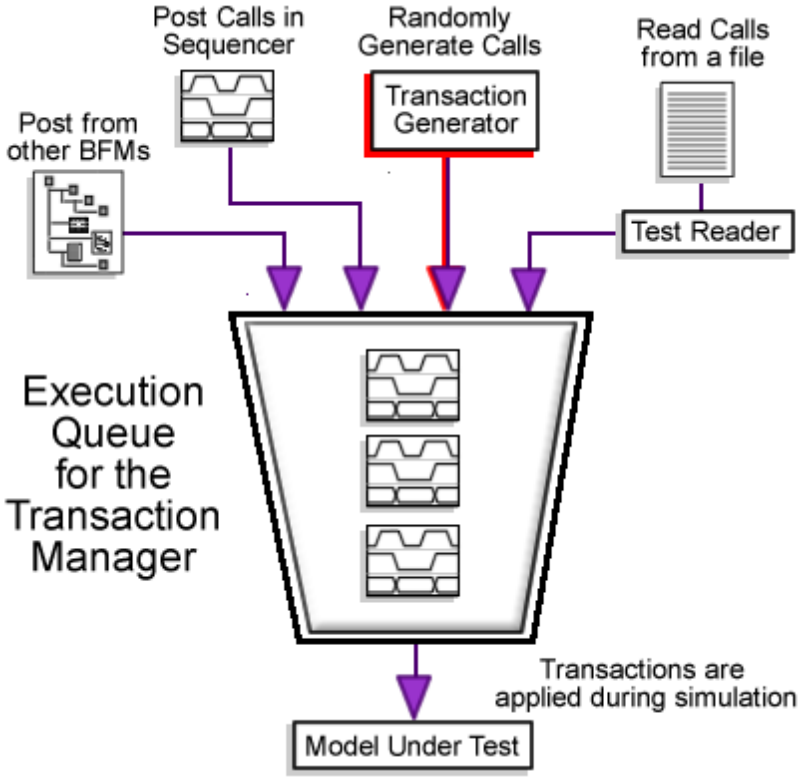
(TBench) 3.14 TestBencher Pro Basic Tutorial Summary

Congratulations! You have completed the TestBencher Pro Basic tutorial. In just a few minutes you created four transaction diagrams, modified the sequencer process, and generated a full bus functional model. This is a very basic example of how TestBencher Pro can work. More advanced features include the ability randomly generate values for transactions or read them in from files. You also extend TestBencher's capabilities by writing functions and calling them within a transaction diagram. The next step is read through the first page of each of the TestBencher Pro Manual's chapters to see the types of functions that are available.

Copyright © 2014, SynaptiCAD Sales, Inc.

Test Bench Generation 4: TestBencher Pro with Random Transactions

This tutorial demonstrates how to convert a directed-test VHDL test bench project into a constrained-random testbench where transactions are randomly applied to the model under test. TestBencher Pro can generate a Transaction Generator that randomly posts master transaction calls to a Transaction Manager Queue, based on the probabilities specified in a weightings table. A weightings table defines the relative probabilities for the next transaction type based on the type of the most recently posted transaction call. The input data for these randomly selected transactions is automatically randomized using the constraint settings for each input variable when a transaction call is popped from the queue and executed. You will need a license to TestBencher Pro and access to a VHDL simulator to simulate and see the results of this tutorial. These features are not currently supported in Verilog.

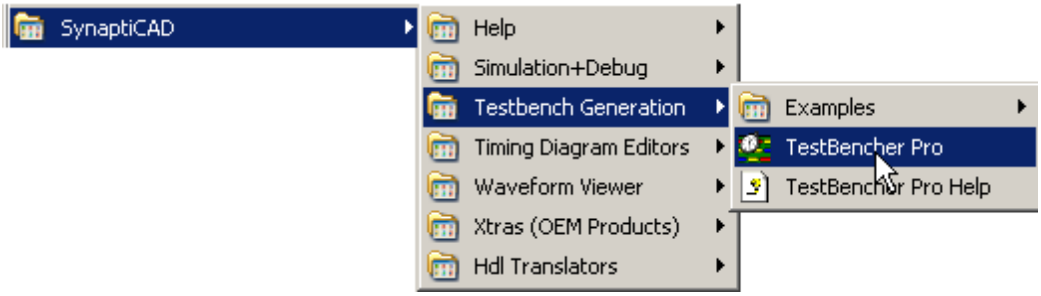


(TBench) 4.1 Run TestBencher Pro

This tutorial requires a full version license for TestBencher Pro or a temporary evaluation license. If you are evaluating then you can obtain a license by completing the form under the **Help > Request License** menu item and contacting our sales department.

- Run TestBencher Pro from the Start Menu.

Copyright © 2014, SynaptiCAD Sales, Inc.

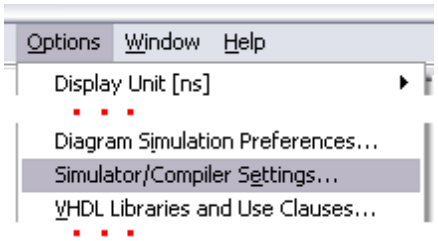


- To verify that you have a license, select the **Help > View License Details...** from the top-level menu of TestBencher to see if your license is detected.

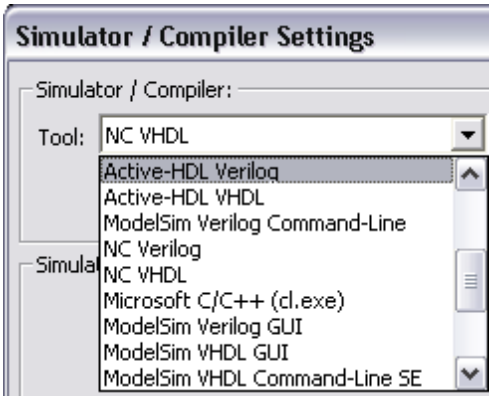
(TBench) 4.2 Setup the VHDL Simulator

This tutorial requires that you have access to a VHDL Simulator from another vendor. You will need to tell TestBencher which simulator you want to use and also recompile the syncad libraries. Each time you upgrade to a newer version of TestBencher Pro or a new simulator you will need to recompile the libraries.

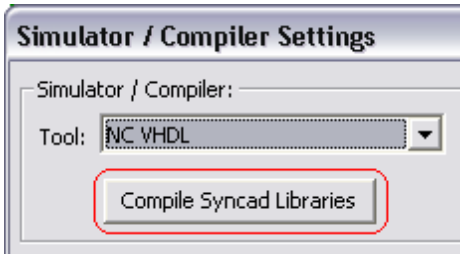
- Choose the **Options > Simulator and Compiler Settings** menu to open a dialog of the same name.



- In the **Tools** drop-down, pick the name of your VHDL simulator.
- TestBencher Pro will look through your normal executable paths to find the stated simulator. There is also a **Simulator Path** box where you can type in the path if it is not in your PATH environment variable's list.



- Press the **Compile Syncad Libraries** button to compile the simulation libraries with the selected simulator.
- Press **Ok** to close the dialog and apply the changes.

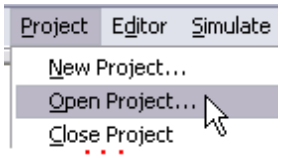


Copyright © 2014, SynaptiCAD Sales, Inc.

(TBench) 4.3 Load the RandomizedSweepTest Project

Load the RandomizedSweepTest project and examine its contents.

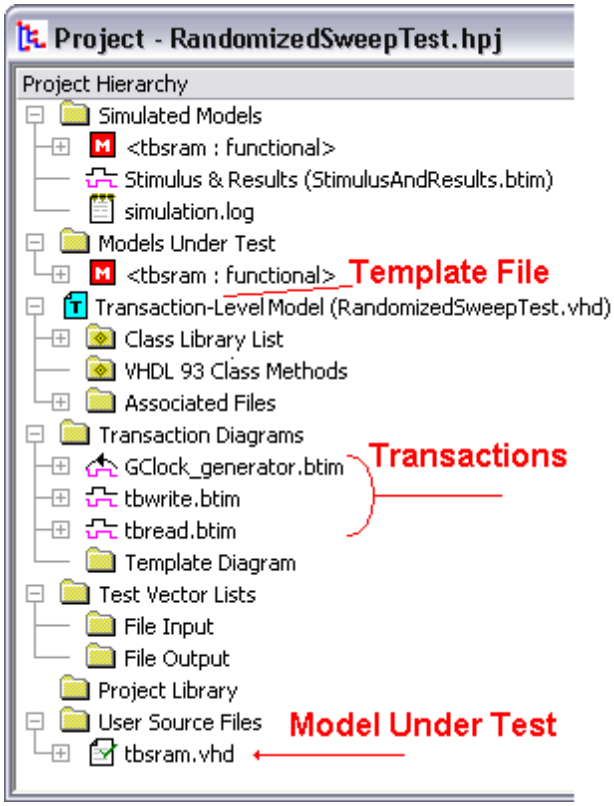
- Choose the **Project > Open Project** menu to open a dialog of the same name.



- Open the **RandomizedSweepTest** project located in the **SynaptiCAD > Examples > TestBencher > VHDL > RandomizedSweepTest** directory.

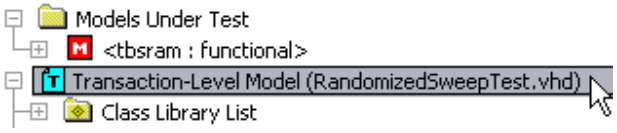


- This project will test the SRAM model located in the **tbsram.vhd** file. Double click on that file to see the code for the SRAM.
- Notice that there are three Transaction Diagrams.
- The **GClock_generator** is a slave diagram that will drive the clock to the other transactions and run in a looping mode.
- The **tbwrite** and **tbread** transactions are master transactions that will run once and stop. They are the write and read transactions that will be used to test the SRAM model under test.
- The **Component Model** contains the template file where the transactions will be sequenced.



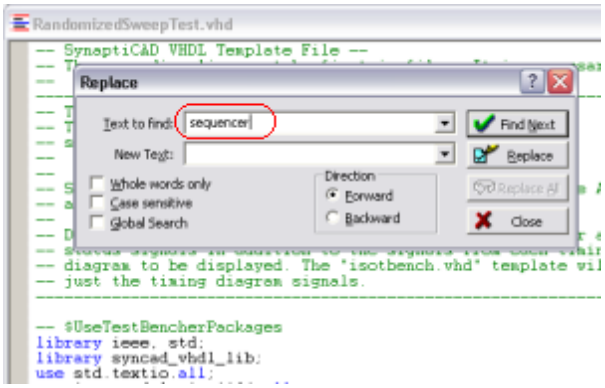
Open the Template file and find the Sequencer Process:

- Open the Template file by double clicking on the **Component Model** in *Project* window.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Find the **Sequencer Process** in the file. We used the **CTRL-F** keys to open the search box and searched for ***sequencer*** to find the process.
- Leave this file open for the next couple of sections.



(TBench) 4.4 Weight the Transaction Types

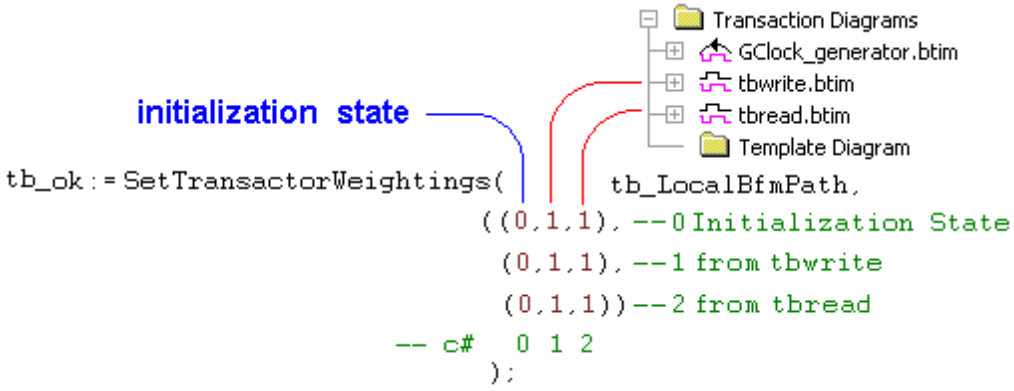
TestBencher uses a weighting table to specify the probability that a particular transaction type will be posted, based on the type of the previous randomly generated transaction type. The default table is set inside the **$InitializeDiagramTaskCall** macro at the top of the Sequencer Process. This default table gives uniform weighting to all master transactions so that each has the same probability that it will execute after any other master transactor. The default table sets the probability of the initialization state to zero (the initialization state is not a transactor type, it just represents the initial state of the BFM before any transactor has been randomly generated, so the first column should always contain zeros).

```
-- Sequencer process
process
 begin

-- ensure all diagrams initialize before applying them
-- $InitializeDiagramTaskCall
tb_initialize_RandomizedSweepTest(tb_ProjectID);
tb_ok := InitializeUniform(7, 7000);
tb_ok := SetTransactorWeightings(tb_LocalBfmPath,
                    RandomizedSweepTestUniformWeighting);
-- End $InitializeDiagramTaskCall
```

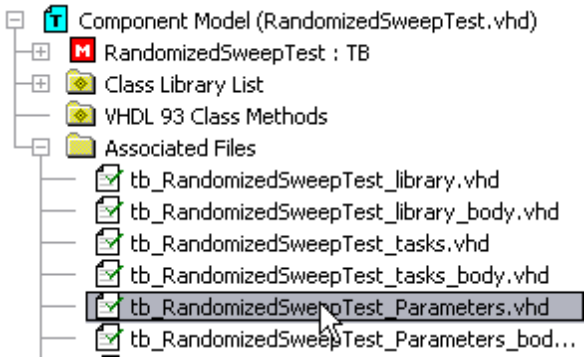
Change the Weighting Function:

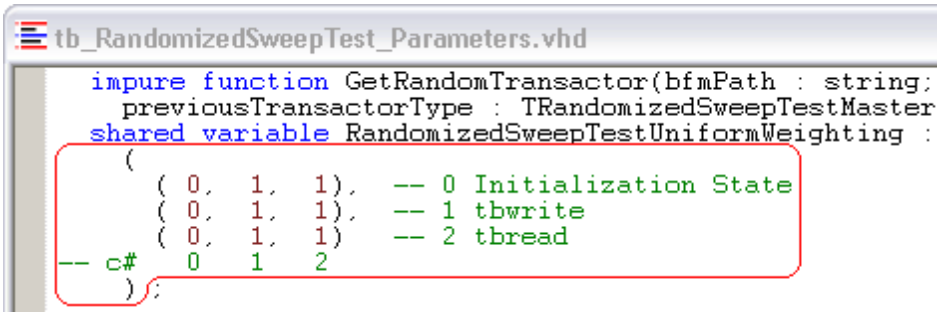
- Copy the entire SetTransactorWeightings line from inside the above macro, and Paste it below the macro. You can then edit this line to have your table values overwrite the default table during simulation.
- The **tb_ok** variable is an automatically generated variable that you can use to capture the return value of TestBencher function calls that return a Boolean value.
- The ***ProjectName*UniformWeighting** term needs to be replaced with a state matrix, where the rows represent the last randomly generated master transactor type and the columns define the weighting for the next transactor type. The first row/column is reserved for the reset or starting state of the BFM. The order of the master transactions is the same as they appear in the Transaction Diagrams folder in the *Project* window.

Copyright © 2014, SynaptiCAD Sales, Inc.



- The table can be typed in or copied from the package file that defines the default weightings table. To copy the table, double click on the **tb_*ProjectName*_Parameters** file located in the **Associated Files** folder of the **Component Model** to view the code.
- Search for **UniformWeighting** in the package file and copy/paste the table as an argument to a **SetTransactorWeighings** call in the sequencer.





- A zero in the weightings table indicates that a specific transaction will never follow another. The higher the number, the more likely a transaction will follow. The table will accept values from 0 to max integer. Here we made it just as likely that a read will follow a write or vise-a-versa by using 1 in all the spaces that do not involve the initialization state. Filling the last column with 2's would make it twice as likely to generate *tbreads* than *tbwrites*.

(TBench) 4.5 Post Random Transaction Types

Calls to **PostRandomTransactionType** will randomly generate one of the master transaction types of the specified BFM and add this transaction to the BFM's transaction manager queue. In converting a directed-test test bench project into a constrained-random testbench (where transactions are randomly applied to the model under test), you will remove most of the apply calls to master transactors and replace them with PostRandomTransactionType calls. The apply calls to slave transactors will probably remain the same.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Scroll down to view the code below the sequencer comment.

```
Apply_GClock_generator_looping_nowait(tb_Control, tb_LocalBfmPath);
for i in 0 to 5 loop
  PostRandomTransactionType(tb_Control, tb_LocalBfmPath);
end loop;
wait for 2000 ns;
Abort_GClock_generator(tb_Control, tb_LocalBfmPath);
```

Start and Stop the Clock

- Notice that the we started and stopped the clock using regular Apply calls. Since the Clock is a slave transaction it is not eligible to be randomized by the Test Generator.
- The **PostRandomTransactionType** call puts one randomly chosen transaction into the Transaction Manager's Queue.
- There is a loop around the **PostRandomTransactionType** call so it will execute and put 5 transactions in the queue.
- There is also a **wait** call that will force the simulation to continue to run until the queue is empty (both the read and the write transaction take 300ns to complete, so 5 of these transactions will complete in 1500ns).

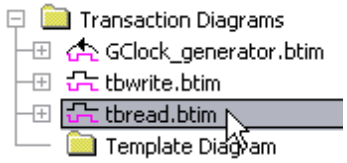
(TBench) 4.6 Constrain the Random Data

Each time a posted random transaction is executed, the input arguments for the transaction will be randomly generated, taking into account any constraint definitions for the transactor's input variables.

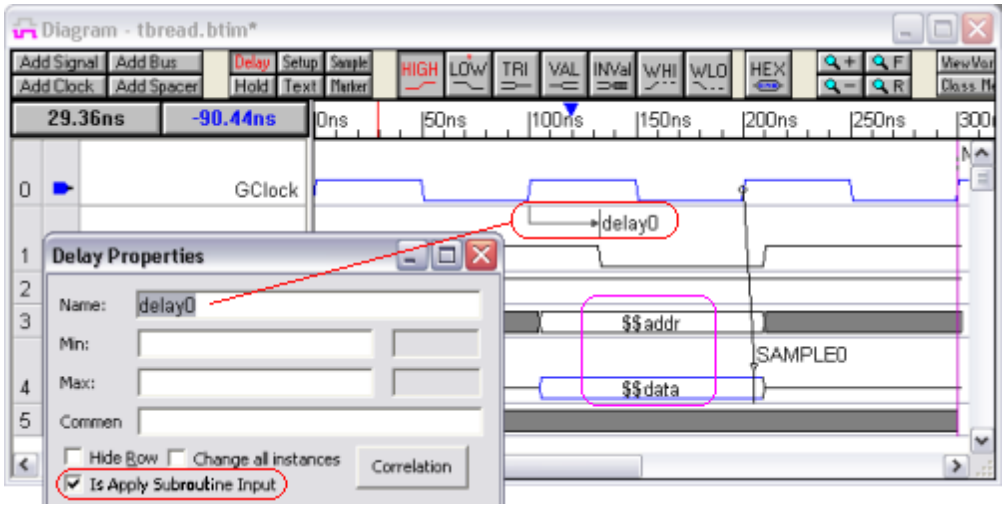
Data into a transaction can also be randomized inside a normal post or apply call by checking the **Randomize Input Parameters** box in the *Insert Diagram Subroutine Call* dialog. It is not necessary to do this when using a **PostRandomTransactionType** call, because the data is automatically randomized into the random transactions. Using either method you will want to constrain the variables using the steps below.

Look at the Diagram Variables:

- Double click on the tbread transactor in the project window to open the diagram.



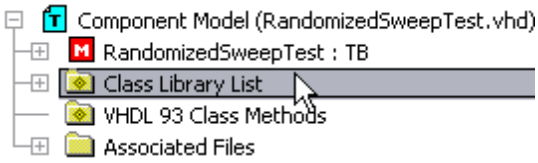
Copyright © 2014, SynaptiCAD Sales, Inc.

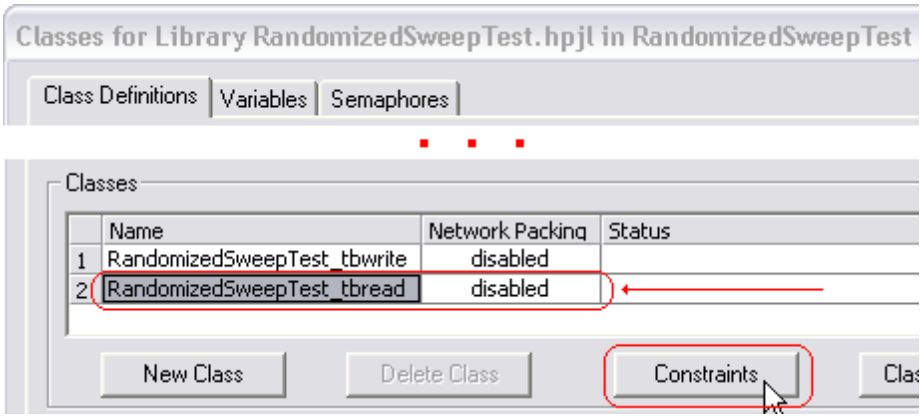


- Notice the **$$addr** and **$$data** variables on the address and data busses. The **$$** means that they are input variables to the diagram. They will be of the same type as the signal they are on. In this case, the variables are on an 8-bit bus of type 4_state, so the variables will have a min/max range of 0-255 and a type of 4_state.
- Double click on **delay0** to open the *Delay Properties* dialog.
- Notice that **Is Apply Subroutine Input** is checked, making the delay value an input to the transactor. Each time this diagram is randomly called, a new delay value will be passed to the transactor.

Constrain the Variables:

- Double click on the **Class Library List** under the **Component Model** branch of the project to open the *Classes for Library* dialog.

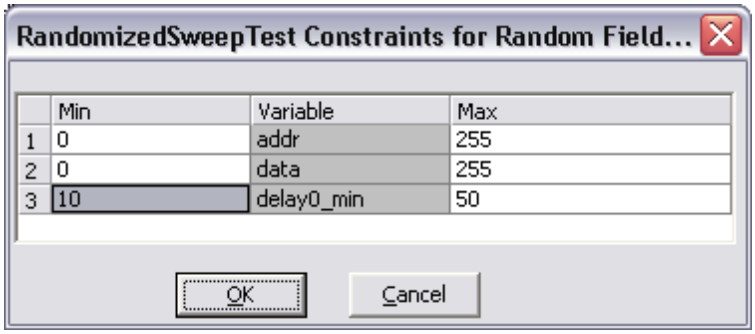




- Select a particular diagram in the **Class Definitions** tab and then press the **Constraints** button to view and edit the constraints. Each transactor diagram has its own set of input variables.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Variables default to the entire legal range for their data types.
- For **addr** and **data,** the entire range of an 8-bit value makes sense for our model.



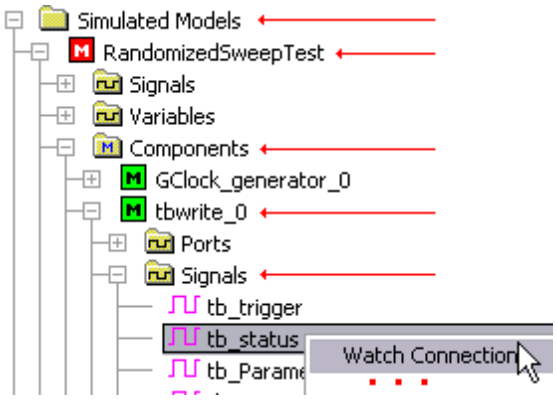
- The **delay0_min** is of type **real**, so many of the legal real values are larger than the possible running times of a transaction, so we constrained the range to 10-50 ns.
- Close all the dialogs when you are done looking at the values.

(TBench) 4.7 Simulate and View the Results

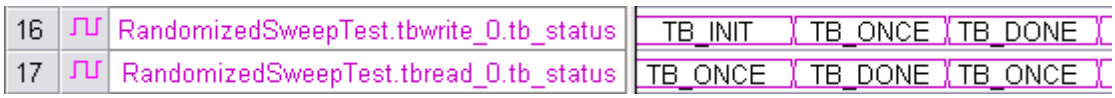
Here we will turn on some status signals to make it easier to tell when a particular read or write transaction is running in the *StimulusAndResults* diagram. Then we will compile and simulate and look at the results.

Set the Status Signals to Watch:

- Open the **Simulated Models** folder and drill down as shown to find the **tb_status** signal of the **tbwrite_0** transactor.
- Right click on the status signal and choose **Watch Connection** from the right click context menu to send this signal to the *Stimulus and Results Diagram*.
- Do the same steps for the **tb_status** signal of the **tbread_0** transactor.



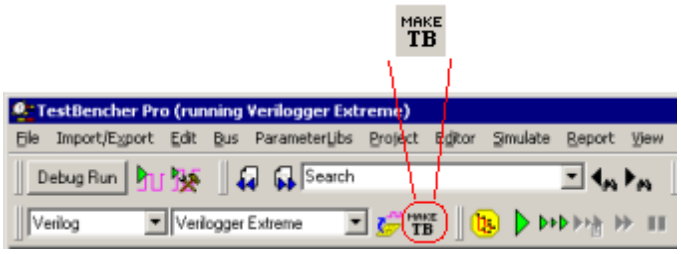
- When you are done, the **tb_status** signals will be displayed in the diagram window. After simulation, you will be able to see the state of the status signals go to **TB_ONCE** when the associated transactor is active.



Copyright © 2014, SynaptiCAD Sales, Inc.

To generate the test bench:

- Press the **Make TB** button. This will expand the macros in the template file and pop up a dialog that says "*Finished generating test bench. Please check waveperl.log for errors*." Close this dialog by clicking the **OK** button.



- In the *Report* window, check the **waveperl.log** tab to see if TestBencher encountered any errors during test bench generation. If it did, fix the error and regenerate the test bench. (If you cannot see the *Report* window, choose the **Window > Report** menu to bring it to the front.)

To simulate the design:

- Click the yellow **Compile Model and Test Bench** button. This builds (parses) the project using the tools specified in the *Project Settings* and *Simulator and Compiler Settings* dialogs.



- In the bottom right corner, a yellow **Simulation Built** status message indicates the build was successful and that you are ready to simulate. If the status indicates an error, the *Errors* tab in the *Report* window will display the compile errors. If there are errors, fix them, regenerate the test bench, and recompile.



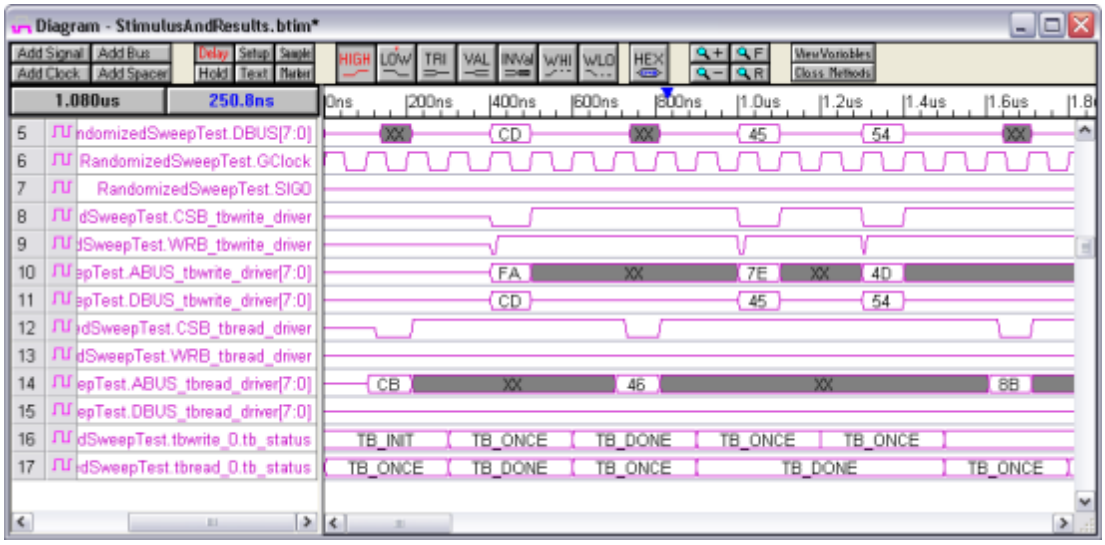
- Click the green *Run* button on the simulation button bar. This will simulate the design and display the results in the *StimulusAndResults* diagram and the *Report* window *simulation.log* tab.



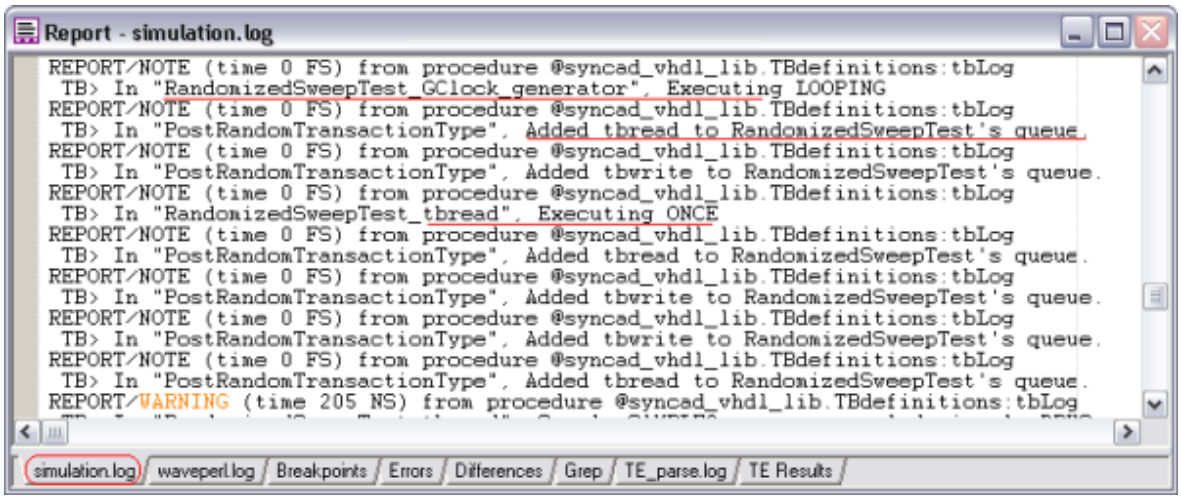
- In the bottom right corner, a **Simulation Good** status message indicates that the simulation has reached a successful end.

View the Results in the StimulusAndResults Window:

Copyright © 2014, SynaptiCAD Sales, Inc.



View the Results in the Simulation Log Window:



(TBench) 4.8 Set the Random Seed

Although the transaction types and transaction input values are randomly generated, you will see the same results each time you simulate, because the seeds for the random number generator are not changed between simulation runs. This is normally desirable, since you want your test bench to generate the same results each time you run it. If you want to change the sequence of randomly generated values, you must change the seed values in the test bench.

Just below the begin statement in the sequencer process is an automatic macro, **$InitializeDiagramTaskCall,** which initializes the transactors of this BFM and also sets the random seed for the BFM's random number generator (this generator is used to randomize the posted transactions and the input data for these randomized transactions). Since this code is automatically generated, you cannot directly edit the seed statement in the macro. However, if you make another call to the **InitializeUniform** function below the macro, your new seed values will overwrite the auto-generated seed values.

Copyright © 2014, SynaptiCAD Sales, Inc.

```
-- Sequencer process
process
 begin

-- ensure all diagrams initialize before applying them
-- $InitializeDiagramTaskCall
tb_initialize_RandomizedSweepTest(tb_ProjectID);
tb_ok := InitializeUniform(7, 7000);
tb_ok := SetTransactorWeightings(tb_LocalBfmPath,
                    RandomizedSweepTestUniformWeighting);
-- End $InitializeDiagramTaskCall
```

This macro is rebuilt each time TB is rebuilt

- To set a different seed, copy the line of code from the macro that calls the **InitializeUniform** function and paste it below the macro end statement. Then edit the seed values passed to this new function call.
- The InitializeUniform function is defined in a package file called ***tbrandom.vhd*** in the syncad_vhdl_library.

(TBench) 4.9 Randomize Transactions Summary

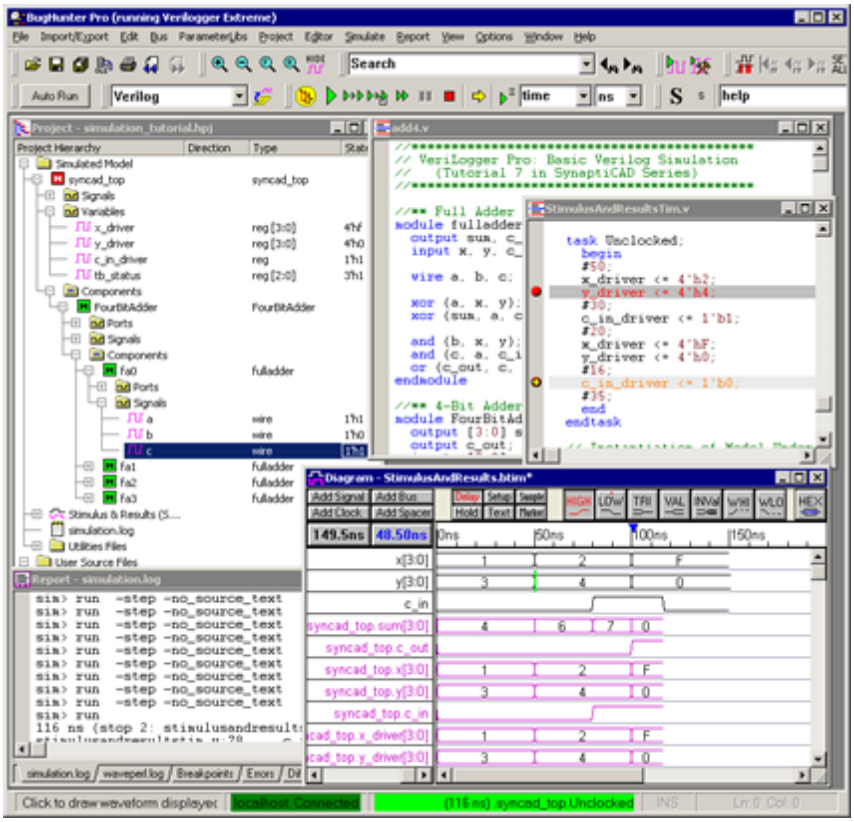
Congratulations, you have completed the Random Transactions tutorial. To convert a regular VHDL project into one that applies random transactions, the following changes were required:

- In the sequencer process, you can make another call to the **SetTransactorWeightings** function call to redefine the random weightings table that specifies the probability that a particular master transaction will be applied after another type of transaction. By default, the transactions will all have the same probability of being generated.
- Probably keep all the slave transaction apply calls the same, because clock processes and slave transactors have to start and stop at particular times in the test bench operation (generally at the beginning and end of the test bench).
- Replace some or all of the master transactor apply calls in the sequencer with **PostRandomTransactionType** calls to randomly post transactions to the BFM's queue.
- Add a **wait** statement to the code or somehow make sure that the simulator will keep running while there are transactions in the queue waiting to be started.
- Constrain the input data going into the transactors by double clicking on the **Class Library List** folder in the Project tree under the Component Model section. Select a transactor diagram, then press the **Constraints** button to view and edit the constraints for that transactor's input variables.
- The **InitializeUniform** function can be called again to replace the default seeds for the random number generator.

Copyright © 2014, SynaptiCAD Sales, Inc.

Simulation 1: VeriLogger Basic Verilog Simulation

This tutorial demonstrates the basic simulation features of the VeriLogger simulators (simx and vlogcmd) and the graphical debugger (BugHunter Pro). It teaches you how to create and manage a project and how to build, simulate, and debug your design. It also demonstrates the graphical test bench generation features that are unique to BugHunter Pro.



This is a stand alone tutorial which you should be able to complete without reading any of the other tutorials. However, if you plan to make extensive use of the graphical stimulus generation features then you may also want to perform the Basic Drawing and Timing Analysis 10 tutorial which covers more of the drawing features of the timing diagram editor.

(Sim) 1.1 Simulator Choices

SynpatiCAD offers two different Verilog simulators: VeriLogger Extreme (simx) and VeriLogger Pro (vlogcmd). VeriLogger Extreme is a high-performance compiled-code Verilog 2001 simulator that offers fast simulation of both RTL and gate-level simulations with SDF timing information. VeriLogger Pro is an interpreted Verilog-95 compliant simulator with a low memory footprint, but it does not support strengths.

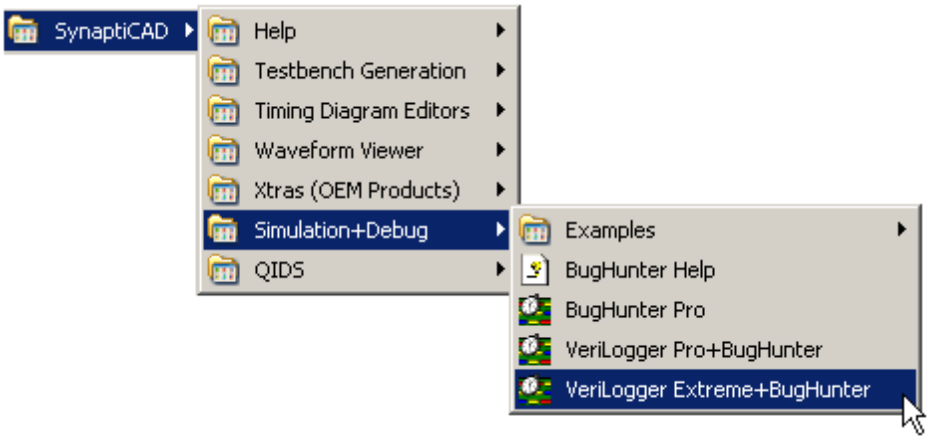
VeriLogger Extreme is the faster simulator for large designs and simulating in **debug run mode** (the standard mode for simulators). However, since VeriLogger Pro is interpreted and does not need to compile the code, it is faster for smaller designs and for **auto run mode** where the user is graphically changing the test bench and kicking off automatic simulations.

When you purchase a simulator, you get SynaptiCAD's graphical debugger, BugHunter Pro, in addition to the command line version of the simulator. Instructions for running the command line

Copyright © 2014, SynaptiCAD Sales, Inc.

versions are found in the **BugHunter and VeriLogger Manual** *Chapter 5: Command Line Simulators*. For this tutorial, run BugHunter so that you can experiment with the graphical debugging interface.

- From the Start Menu, choose one of the VeriLogger simulators running under BugHunter.



- An alternative way to launch the simulators is using the command line:
  - VeriLogger Extreme: **syncad -p bhp -s verilogger_extreme**
  - VeriLogger Pro: **syncad -p bhp -s vlogcmd**

  The -p bhp option says to run the BugHunter Pro product. The -s option sets the default simulator used for new BugHunter projects. Note that the simulator used by a project can be changed at any time from inside BugHunter by selecting the **Project > Project Simulation Properties** menu option and changing the the **Simulator Type** under the **Verilog** tab in the dialog that appears.

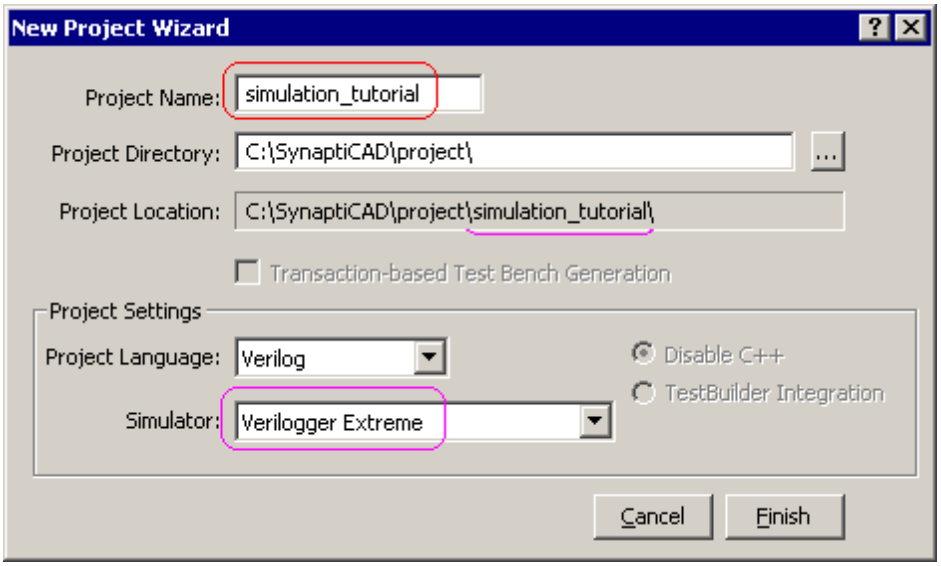
(Sim) 1.2 Add Files to the Project

BugHunter uses a project file to list the files to be simulated and store the simulation options. Here you will create the project file and investigate the source code that is used in the tutorial.

Create a new Project:

- Select the **Project > New Project** menu to open the *New Project Wizard* dialog.
- In the **Project Name** box, name the project **simulation_tutorial**. Notice that the wizard is also creating a directory of the same name as you type in the project name.
- Notice that you can select the simulator used by the project by changing the selection in the **Simulator** drop down box at the bottom of this dialog.
- Press the **Finish** button to create the project.

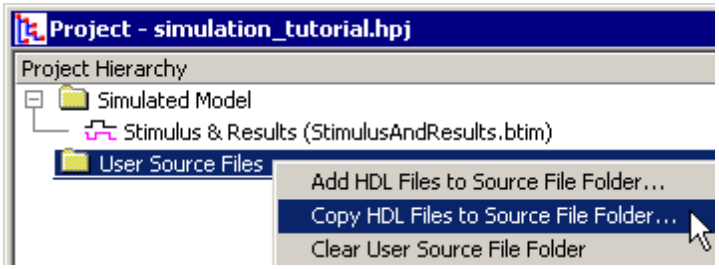
Copyright © 2014, SynaptiCAD Sales, Inc.



Add the Source Files to the Project:

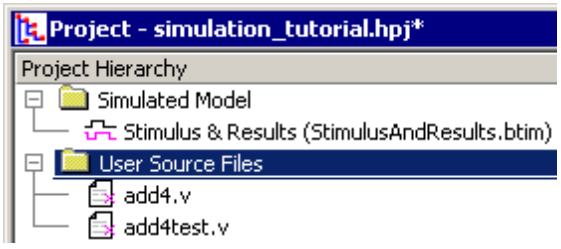
If you were starting a design from scratch, you could use the **Editor > Open HDL File** to open an editor window and type in the code, and then ***add*** the new file to the project. In this tutorial, we will just ***copy*** existing source files from the examples directory.

- Right click on the **User Source Files** folder and select **Copy HDL files to Source File Folder** from the context menu to open a file dialog.



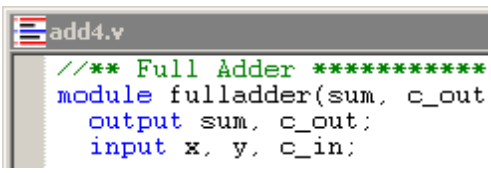
- In the file dialog, browse to the **SynaptiCAD\Examples\TutorialFiles\VeriloggerBasicVerilogSimulation** directory and select the **add4.v** and **add4test.v** files. To select multiple files, hold down the **<CTRL>** key while selecting files with the left mouse button. Then close the file dialog.

- When the files are first added to the project they will be marked with a purple x to show that the files are not compiled.



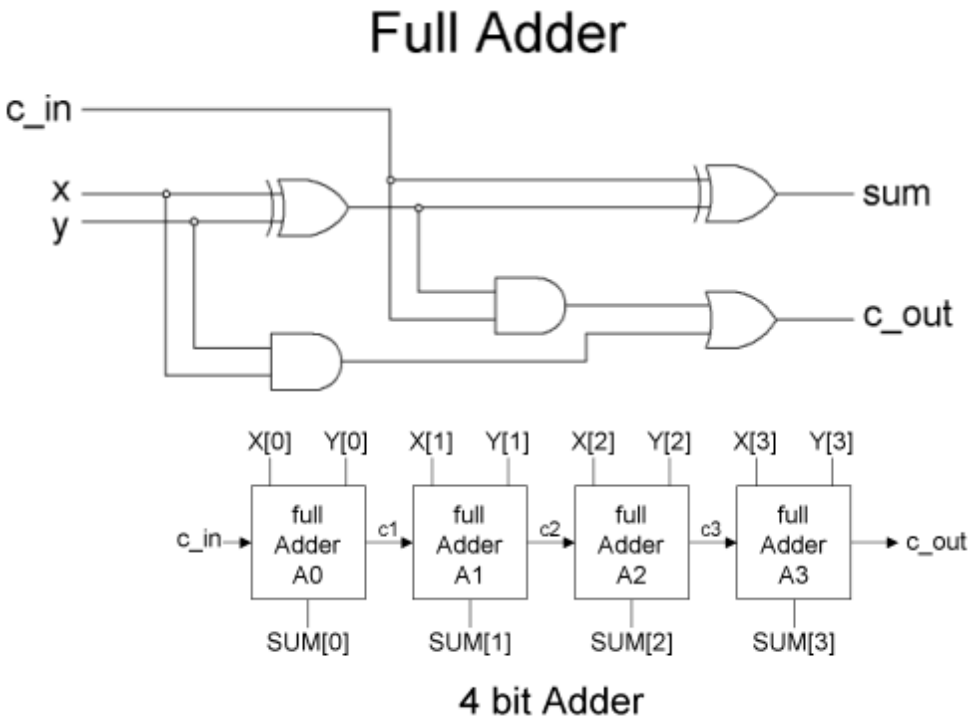
Investigate the code for the Tutorial:

- In the project window, double click on add4.v to launch an editor window with the source code loaded.



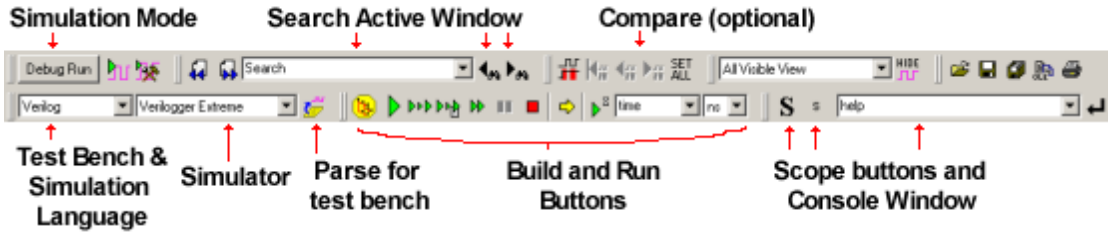
Copyright © 2014, SynaptiCAD Sales, Inc.

- Look through the code and compare it to the following schematics. We will be simulating a 4-bit adder circuit which adds the **x** and **y** inputs together and outputs the answer on the **sum** and **c_out** lines.



(Sim) 1.3 Build the Tree and Investigate the Project

In this section we will build the project tree and use the tree to view the internal modules. Before we begin, take a look at the Simulation Button Bar and familiarize yourself with the buttons. This button bar will control most aspects of the simulation. Slowly pass the mouse over the buttons and read the tool tips.



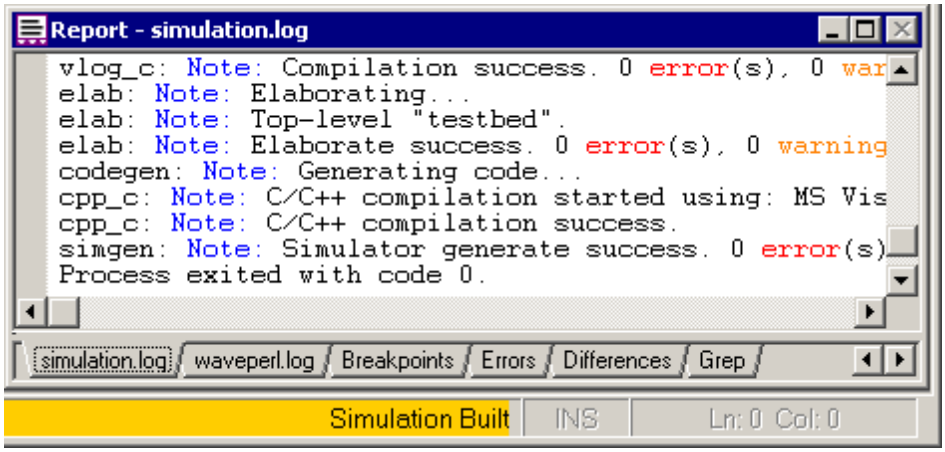
Three ways to build a project:

- Compile the project by pressing the yellow **Build** button on the simulation bar, selecting the **Simulate > Build** menu, or pressing the **<F7>** key.



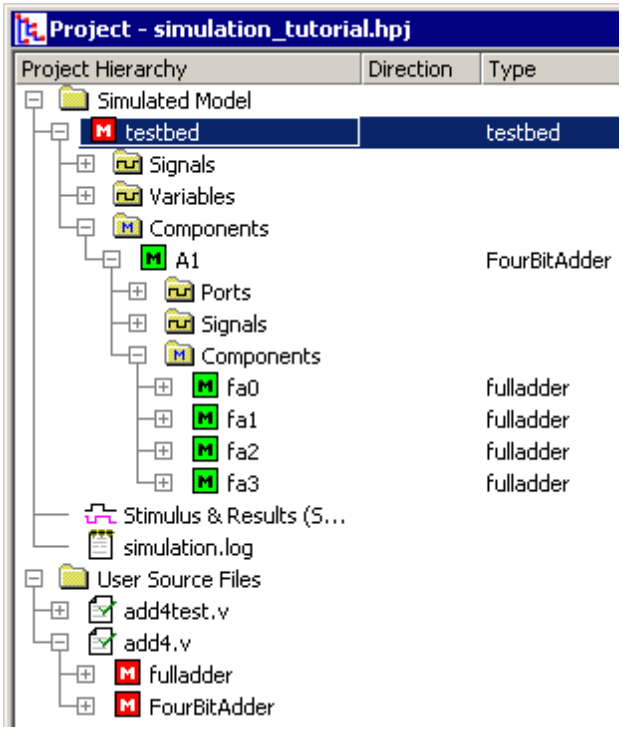
- When the build is successful, a yellow **Simulation Built** message will display in the lower right hand corner of the program.
- The *Report* window **Simulation Log** tab shows the build results, and the **Error** tab would display a hyper-linked list of errors if there were any.

Copyright © 2014, SynaptiCAD Sales, Inc.



Setting Top-level instances after the first Build:

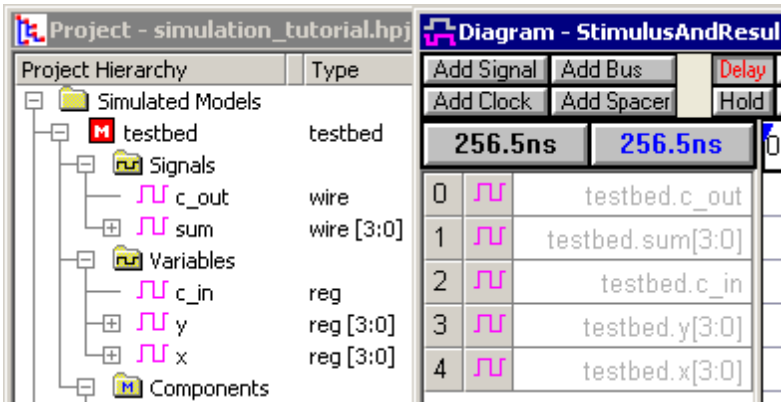
- By default, BugHunter identified **testbed** as the only top-level instance, because all other modules are instantiated under it. For this tutorial, this is the desired top level instance.
- After the first build, you can optionally right click on other modules and choose **Set as a Top Level instance** to force instantiation as top level instances.
- Notice that the User Source files have green check marks to indicated that they are built.
- Double clicking on any component will open a editor scrolled to that place in the code.



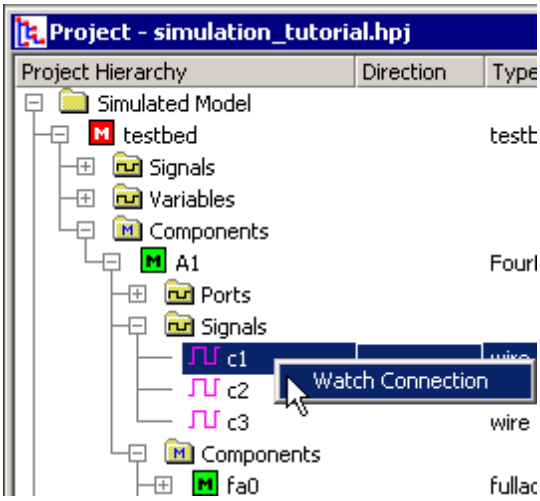
Set Watch Signals after the first build:

Copyright © 2014, SynaptiCAD Sales, Inc.

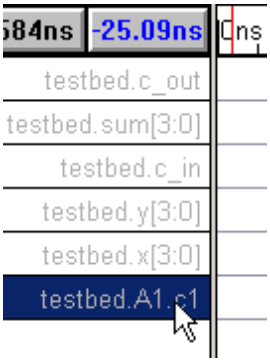
- BugHunter by default sets *watches* on all the signals and variables of the top-level module. This means that the signal names are displayed in the **Stimulus and Results** diagram and the waveforms will be displayed during simulation. Event history is only maintained for these signals.
- The purple waveform icon means the signals are internal to the model and are not ports.



- Open the project tree until you find ***component A1*** and ***signal C1***, then right click and choose the **watch** menu option. This causes C1 to be added to the **Stimulus and Results** diagram.
- You can also set watches on entire components, blocks or variables using the same technique.



- Remove any extra watch signals that you added in the last step. First left click on the signal name in the diagram window to select it, then press the **<delete>** key.



Copyright © 2014, SynaptiCAD Sales, Inc.

(Sim) 1.4 Simulate the Project

The green buttons on the Simulation button bar cause the simulator to run. The smaller buttons are for single stepping through the code. The one with the hourglass will simulate for a specified period of time.

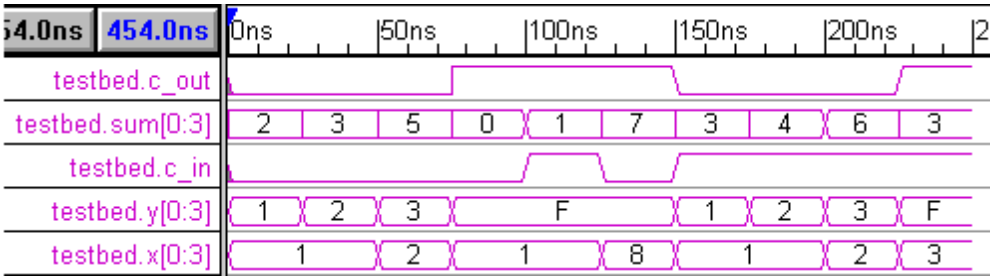


Simulate the project:

- Press the large green run button to simulate until the end of the simulation or until a breakpoint is encountered.



- Notice that the **Stimulus and Results** diagram has displayed the simulation results. Verify that **sum** and **c_out** are correctly being computed as x + y + c_in.



- If there were errors in the code, they would be indicated by the status bar in the lower right hand corner, and listed in the *Report* window's **Error** tab.

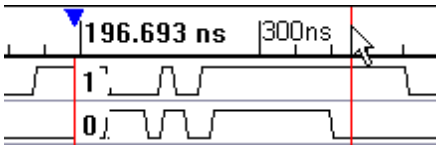
For the rest of this section, just play around with the zooming, scrolling and searching capabilities of the waveform window.

Zoom using buttons or the mouse:

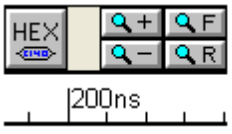
- To zoom in and out quickly, hold down the **<Shift>** key while using the **scroll wheel** on your mouse.

<Shift> and mouse **scroll wheel**

- To zoom in over a visible section, drag and drop inside the **Time Line.**



- The zoom buttons are located on button bar in the diagram window or 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.



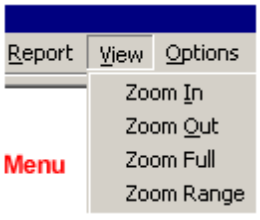


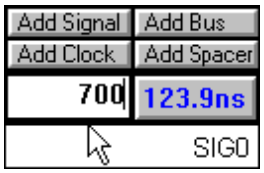
Diagram Window or Menu

- 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.

Copyright © 2014, SynaptiCAD Sales, Inc.

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.



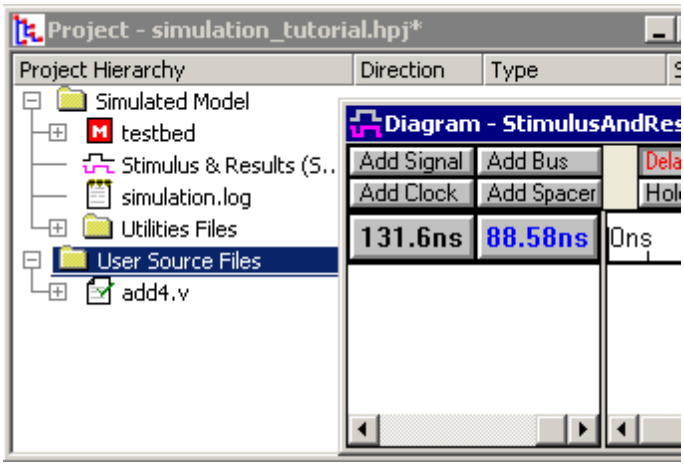
- If the Diagram window is selected, then it will search for either signal names or waveform state variables depending on the object that is selected within the diagram.
- If the Parameter window is selected, then it will search for parameter names.
- If the Report window is selected, then it will search for text in the selected report tab.

(Sim) 1.5 Prepare for Graphical Test Bench Generation

So far in the tutorial, we have created a project and simulated some code using a manually written testbench, which is the traditional design flow using a simulator. In the next few sections, you will be drawing a testbench using SynaptiCAD's graphical testbench generator. Before we start, we must prepare the project by removing the manually written testbench file and clearing out the Stimulus and Results diagram. Then extract the MUT ports into the Stimulus and Results diagram.

Remove the Test Bench Source Code file and empty the Stimulus and Results Diagram:

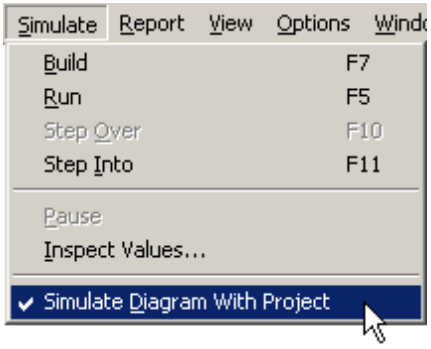
- Select the file **add4test.v** file in the project window and press the **<delete>** key.
- Delete all of the signals in the **Stimulus and Results** diagram by selecting the signal names and pressing the **<delete>** key.
- Verify that only **add4.v** is listed on the project tree, and that the diagram is empty.



Copyright © 2014, SynaptiCAD Sales, Inc.

Verify that BugHunter is in the proper mode to generate a test bench:

- Verify that the **Simulate > Simulate Diagram With Project** menu option is checked. This option lets the simulator compile both the drawn waveforms and the Verilog source code files together. If this is unchecked, then no testbench will be created.



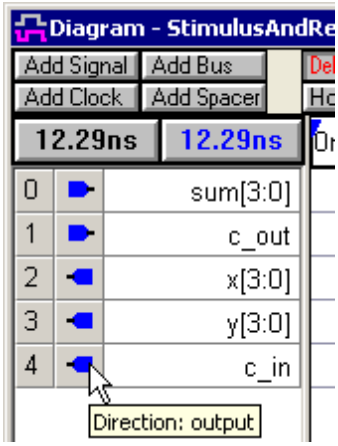
Extract Ports from the Model Under Test (MUT):

In the previous section, the **Stimulus and Results** diagram displayed only signals output by the simulator (the results). This diagram can also hold a testbench that will exercise the model under test (the stimulus). First we will extract the ports from the model under test and later we will draw the test bench.

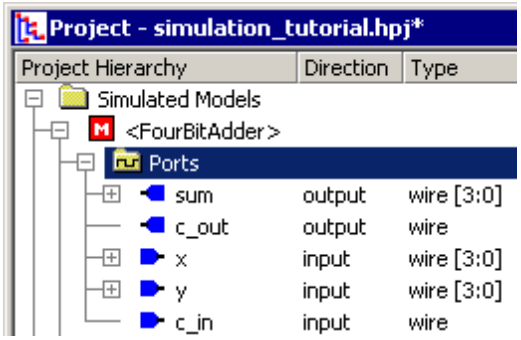
- Press the **Extract the MUT ports into Diagram** on the simulator button bar.



- Notice that the ***Stimulus and Results*** diagram is populated with the ports of the **FourBitAdder** module (see project tree below). These will be the signals that you will draw on in the next section.
- The blue icons pointing to the right are inputs to the test bench (outputs of the model under test). The icons pointing to the left are outputs of the test bench (stimulus that drives the model under test). The tool-tip will always show you the direction if you forget.



- The Extract the MUT function makes a guess as to which model is the model under test and displays that model with single brackets, <>, underneath the **Simulated Model** folder. It guessed correctly for this tutorial.



- If you wanted to pick a different model under test, right click on a different model under the **User Source Files** list and pick **Set as Model Under Test** (don't do this for this tutorial).

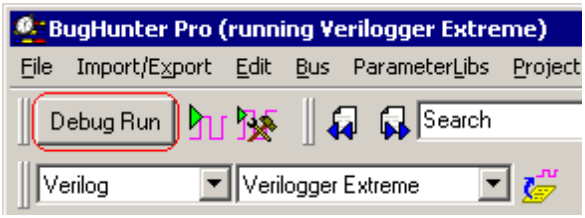
Copyright © 2014, SynaptiCAD Sales, Inc.

(Sim) 1.6 Draw Test Bench in Debug Run Mode

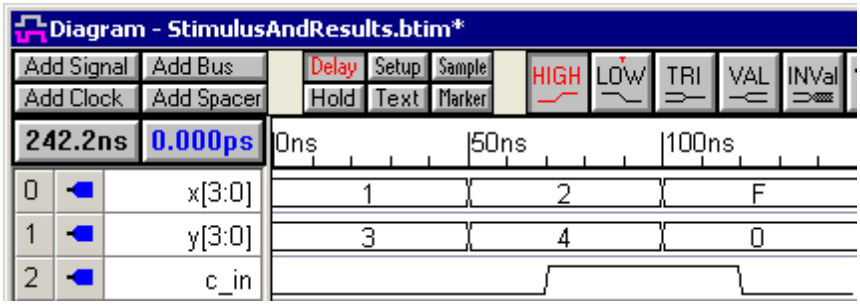
Draw the test bench on the input ports of the of the MUT. You can draw anything and see the results, however the result images shown in this tutorial will be using the timing diagram shown below.

Draw the Test Bench

- Make sure the simulation mode is set to **Debug Run**, rather than **Auto Run**, so that the simulator does not re-simulate while you are drawing.



- Draw the following waveforms on the ***input ports*** of the MUT (see below for drawing instructions if you need them). Notice that the waveforms are ***black*** to indicate that will drive the MUT. If you have never used SynaptiCAD's drawing environment, then read the rest of the instructions in this section and practice drawing random signals before drawing the following diagram:



- Notice also that if you draw on the ***output ports*** (*sum* or *c_out)* the waveforms would be ***blue*** to indicated that they are *expected outputs* from the MUT. This expected output is used by the *Reactive Test Bench Option* to create self-checking test benches, which is covered in the Reactive Test Bench Option Tutorial 122. For this tutorial, you can delete the **sum** and **c_out** signals because we will not use them in this tutorial.

Basic Drawing Instructions (Skip to Section 1.7 if you can draw the timing diagram)

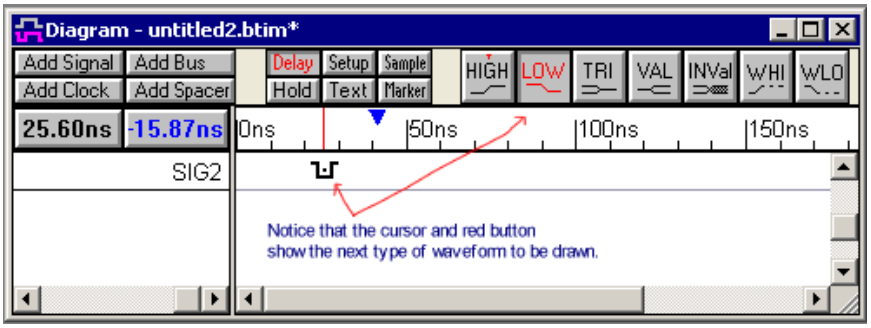
The timing diagram editor is always in drawing mode, so left clicking on a signal will draw a waveform. The red state button controls the type of waveform that is drawn (high, low, tri-state, valid, invalid, weak high, and weak low). The buttons toggle back and forth between two states, and the next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



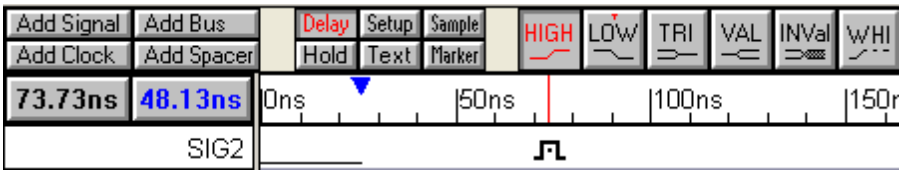
To draw the waveform of a signal:

- 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

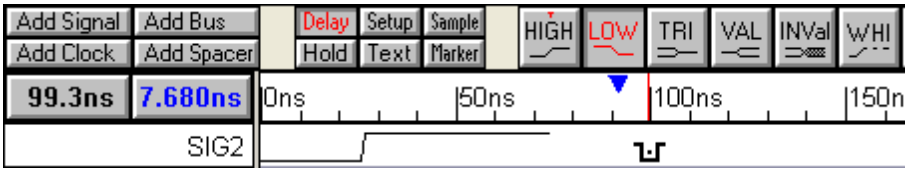
Copyright © 2014, SynaptiCAD Sales, Inc.



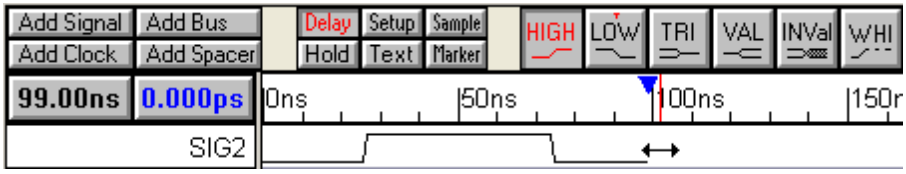
- Click the left mouse button. This draws a waveform from the end of the signal to the mouse cursor.



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



- Keep drawing from left to right across the diagram.

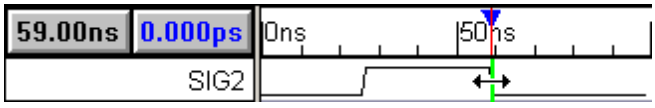


- Pressing the **middle mouse button** either toggles the state buttons or cycles through them depending on the setting in the *Design Preferences* dialog. Choose **Options > Design Preferences** menu to open the dialog.

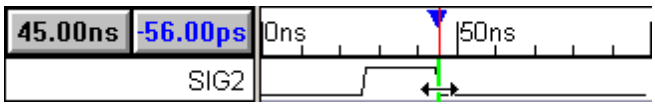
There are several mouse-based editing techniques used to modify existing waveforms. These techniques will only work on signals that are drawn. They will not work on generated signals like clocks and simulated (purple) signals.

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.



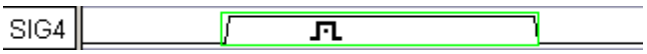
Copyright © 2014, SynaptiCAD Sales, Inc.

- To move transitions on different signals, first select the transitions by holding the **<CTRL>** while clicking on them. Then drag the transition to desired location.



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

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

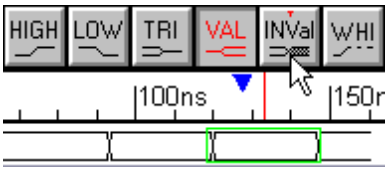


- The inserted state is determined by the red state button

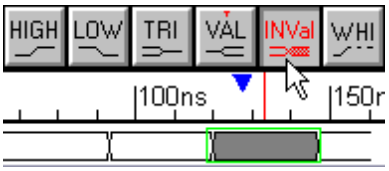


3) Change a segment's graphical state by selecting it and then pressing a state button:

- 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

- 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.

(Sim) 1.7 Simulate in Auto Run Mode

In this section, you will build and simulate the MUT with the graphical testbench. Then you will experiment with the Debug Run and Auto Run simulation modes.

Build and Simulate the Project:

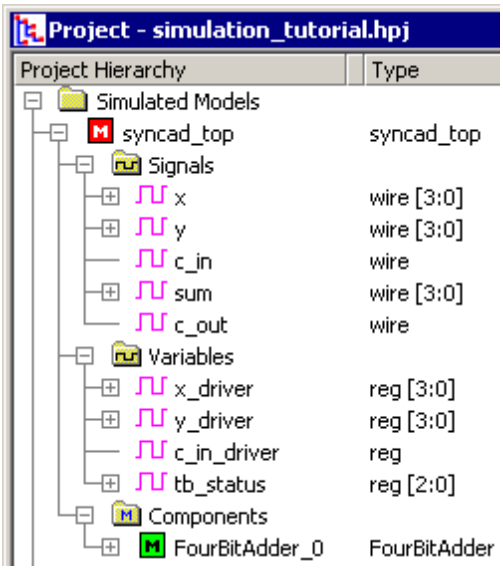
- Press the yellow **Build** button on the simulation bar.



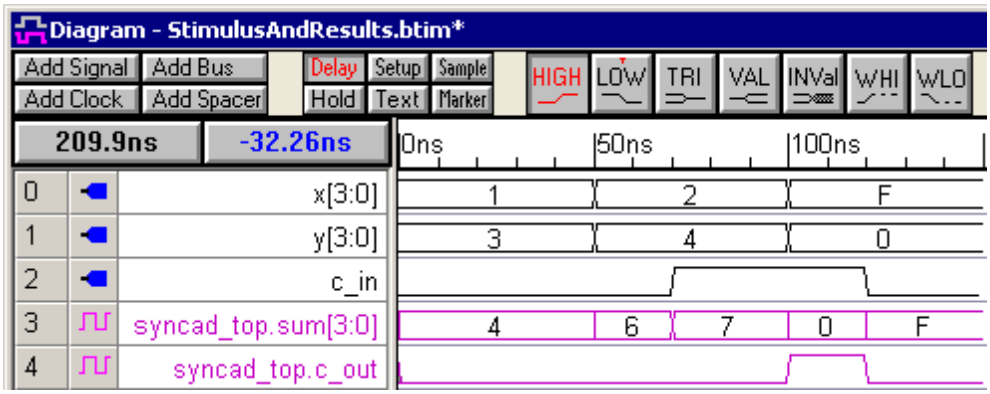
- Notice that after the build is complete, a new model called **syncad_top** was added to the *Simulated Models* tree. This is the top-level

Copyright © 2014, SynaptiCAD Sales, Inc.

instance for the project and it hooks up the graphical test bench signals to the model under test (FourBitAdder). Double clicking on syncad_top will let you view the generated test bench code.

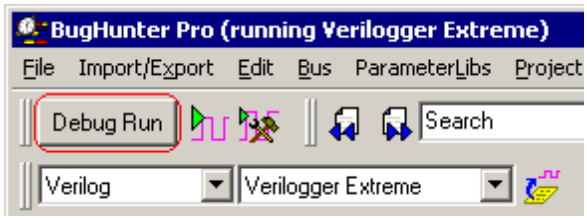


- Also notice that all of the signals and variables of syncad_top have been added to the **Stimulus and Results** diagram (like in section 1.3).
- For this tutorial we are concerned with the outputs of the MUT, so you can ignore all of the signals except the **sum** and **c_out** signals.
- The **driver** and **status** signals are used by our TestBencher Pro product to control the execution of multi-diagram testbenches. These are not used in the tutorial.
- Press the large green **Run** button to simulate the entire test bench. 
- Verify that **Sum** and **c_out** equals **x + y + c_in**. The schematic is shown in section 1.2 175. In the diagram below we dragged and dropped the sum and c_out signals to the top of the simulated signals to make a smaller picture.



Debug Run versus Auto Run Simulation Modes:

- Currently the simulator should be in **Debug Run** mode, so that simulations are only compiled when the Build button is pushed.



- Drag and drop an edge on **c_in** and notice that the simulated waveforms do not change. To update the waveforms, press the **Build** button, followed by the **Run** button to update the simulation output.  and 
- Press the **Debug Run** mode button to toggle the mode to **Auto Run**. 

Copyright © 2014, SynaptiCAD Sales, Inc.

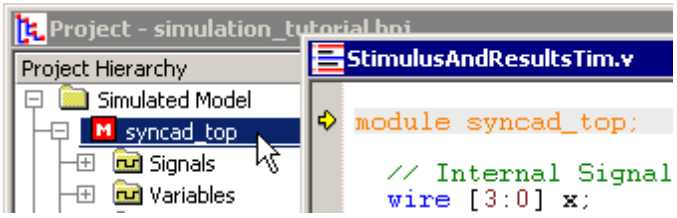
- Drag and drop the same edge on **c_in** and now notice that the simulated waveforms change each time an input is changed. In Auto Run mode, simulations are performed each time a waveform is moved (the Build and Run actions are automatically performed). However, if the simulator is paused in the middle of a simulation when the waveform is changed, then you must manually restart the simulation to apply the change. This keeps minor mouse clicks from prematurely exiting a debug session.
- Experiment with dragging edges and changing the values of the virtual states. If this was a low-level module that you just designed, then you could quickly check the functionality of the module without having to design a formal test bench. If you are running VeriLogger Extreme, you need to wait between changes for the simulator to compile and finish the previous simulation.

(Sim) 1.8 Breakpoints, Stepping and Inspecting

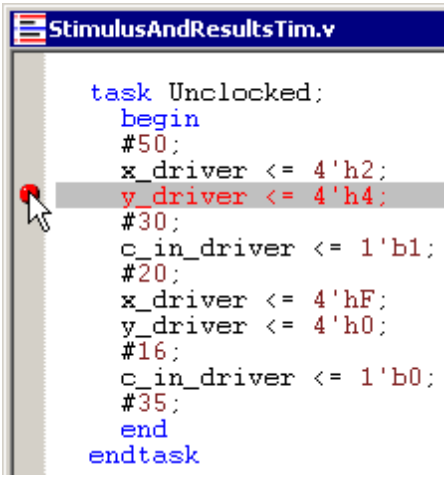
In this section we will place a breakpoint in the generated code and take a brief look at the debugging environment.

Insert a code breakpoint and simulate

- Double click on the **syncad_top** module in the project tree to open an editor that displays the generated code.

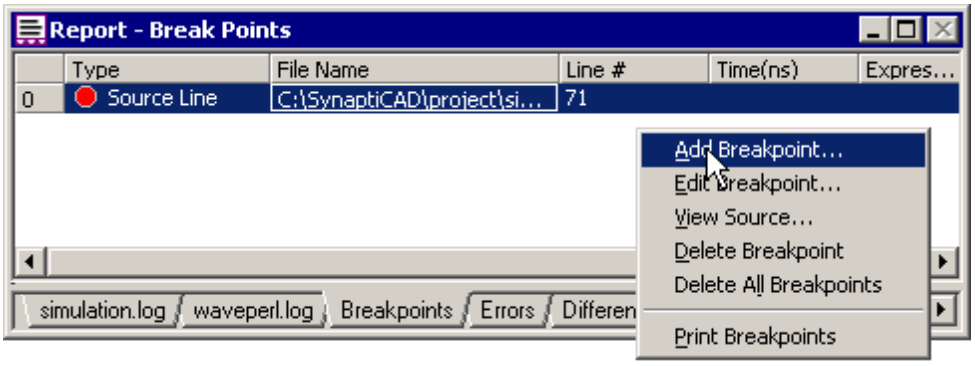


- Scroll down to the **task Unclocked** module, to see the stimulus code that was generated from the drawn waveforms.
- Place a breakpoint on the first change of the y_driver signal by left clicking on the grey bar on the side of the editor window. This will place a red dot in the margin.



- Also note that the *Report* window **breakpoints** tab lists the source line break point. BugHunter also supports time and condition breakpoints that are covered in Chapter 3 Debugging of the BugHunter and VeriLogger manual. These other kinds of breakpoints can be added by right clicking on the breakpoint tab window and choosing **Add Breakpoint** from the context menu, but the easiest way to add a condition breakpoint on a signal is to right click on it in the project tree and select the **Add/Toggle Condition Break Point...** menu option. This will cause the simulation to stop whenever the signal changes value.

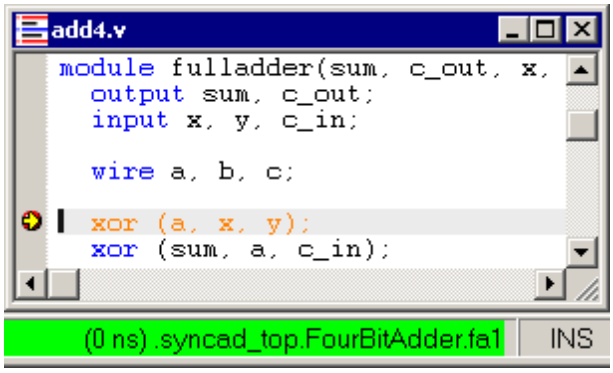
Copyright © 2014, SynaptiCAD Sales, Inc.



- Press the large green **Run** button to simulate to the breakpoint.This particular code line will execute twice at time 50 because it is a non-blocking statement. The expression values of Non-blocking statements are evaluated when the statements are first encountered, but they only update their assigned signal at the end of the simulation cycle (as opposed to blocking statements which evaluate and immediately update the assigned signal).



- Add a breakpoint to the first ***exclusive or*** in the **add4.v** file, then restart the simulation.
- Press the run button a few times and watch the green status bar at the bottom. The fulladder model is instantiated 4 times inside the FourBitAdder module, so you are going to hit this breakpoint a lot. The status bar shows which instance is executing. Here, the **fa1** instance of **fulladder**, which is instantiated in module **FourBitAdder** is about to execute.



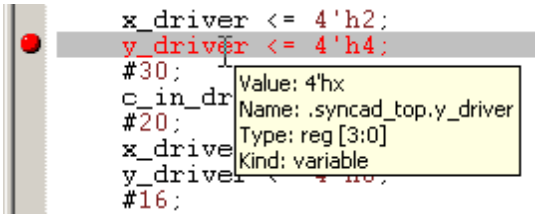
Single step through the code

- Next press one of the single step buttons a couple of times to watch the execution of the code. Notice that **step with trace** (the middle button) leaves a trail of statements in the simulation log tab of the *Report* window. Make sure to check the status bar and compare the execution to the schematic in section 1.2 175.



Use the Inspect Values

- Put the cursor over a variable that has been initialized to see its current value.



- Variables that are watched in the **Stimulus and Results** diagram can be inspected for values

Copyright © 2014, SynaptiCAD Sales, Inc.

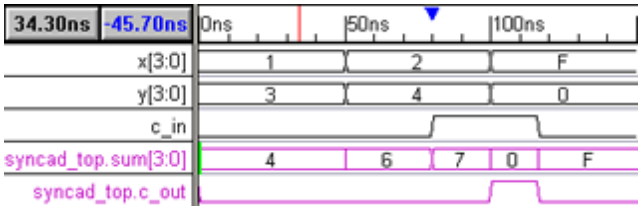
at previous times. See the instructions in the **BugHunter and VeriLogger** manual for using the **Simulate > Inspect Variables** menu function (to fill the window, drag signal names from the **Stimulus and Results** diagram or type them in manually).

(Sim) 1.9 Archiving Stimulus and Results

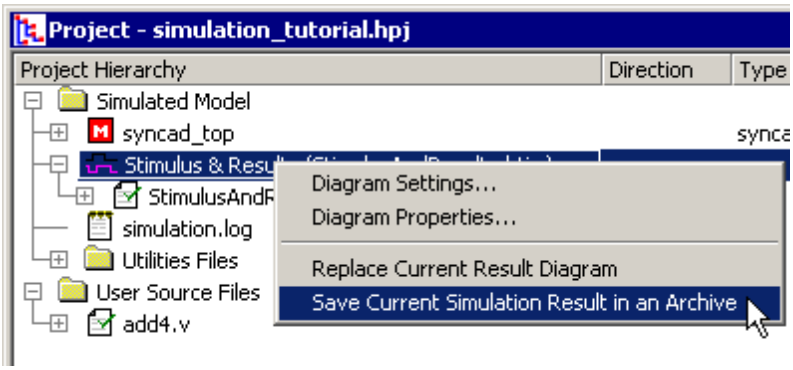
So far we have only used a single **Stimulus and Results** diagram, however multiple diagrams can be switched in and out to test different aspects of the design, or archived off to be used as comparison diagrams for later simulations.

Archive off the simulation results:

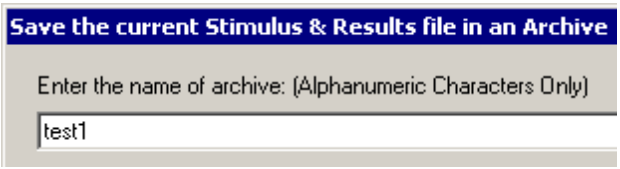
- Run the simulator to the end of the simulation so that the Stimulus and Results diagram and simulation log file are full of data.



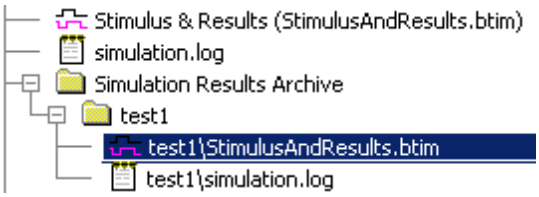
- Right click on the Stimulus and Results node and choose **Save Current Simulation Result in an Archive** from the context menu, to open a dialog.



- Name the archive **test1** and close the dialog. This will create a subdirectory called test1 under the project directory.

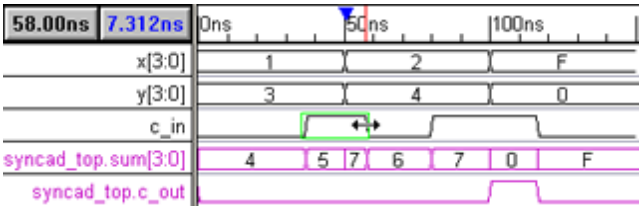


- Notice that a new node called **Simulation Results Archive** has been added to the tree. Both the Stimulus and Results diagram and the simulation log file have been copied to the new archive directory.



Create a new test and archive that off:

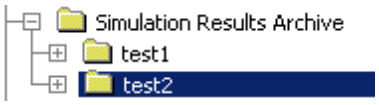
- Modify the stimulus and then simulate to see the changed waveforms. If you are in Debug Run mode you will have to press the Build then the Run button to restart the



Copyright © 2014, SynaptiCAD Sales, Inc.

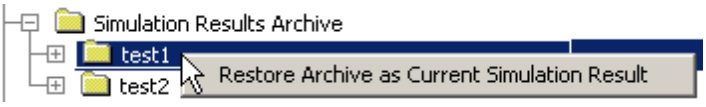
simulation.

- Archive these results to an archive named **test2** using the method described above.

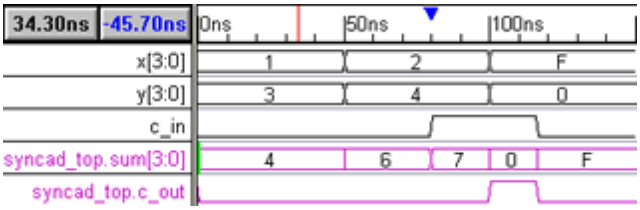


Switch back to the original test:

- Right click and choose **Restore Archive as Current Simulation Result** from the context menu to restore the results



- Notice that the **Stimulus and Results** diagram has been replaced with the original test1 data



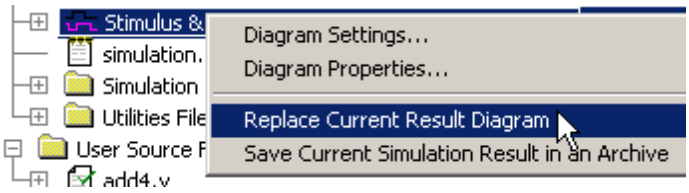
- The archive files remain untouched by the **restore** command. The data is copied from the archive into the working directory and you need to re-archive if you want to save the new data.
- If you purchase the **Compare Option**, then you can use the archive diagrams to compare against new simulations of your design.

Multiple Stimulus and Results diagrams:

In the previous example, both archives used the default **Stimulus and Results** diagram name so it can get a little confusing. However, the diagram can be named any name.

- Save the **Stimulus and Results** diagram to a new name using the **File > Save Current Diagram** menu option.

- Right click on the **Stimulus and Results** node and choose **Replace Current Result Diagram** from the context menu, then choose the file that you saved.



(Sim) 1.10 Saving the Project files

Whenever you build a project, the project file and any modified source files are automatically saved. You can also manually save files at any time. This section describes how the different file types can be saved manually.

Save the HPJ project file:

- Choose the **Project > Save Project** menu option to save the *.hpj file. The project saves the simulation options and the names of the files contained on the project tree. It does not save the source code or the watched signals.

Copyright © 2014, SynaptiCAD Sales, Inc.

Save the Source code:

Each time you simulate, every open editor is queried to determine if the source code needs to be saved before the simulation starts. You can also force a save by doing the following:

- Select the **Editor > Save HDL File** menu option to save the source code in the editor with the focus.
- Select the **Editor > Save All** menu option to save the source code in all opened editors.

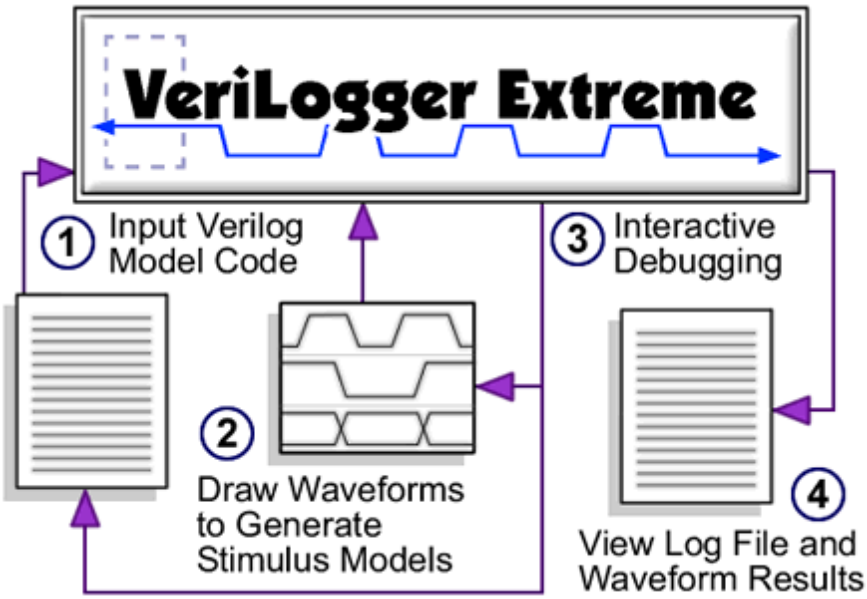
Saving the Stimulus and Results diagram:

The watch signals and simulation results are saved in the current stimulus and results file.

- Click on the **Stimulus and Results** diagram window, then select the **File > Save Timing Diagram** menu option to save the diagram. Note: the evaluation version does not allow diagrams to be saved (you will need to buy a full license or get a temporary evaluation license to perform this function).

(Sim) 1.11 Summary of VeriLogger Basic Verilog Simulation

Congratulations, you have completed the VeriLogger Basic Verilog Simulation tutorial. We have demonstrated how to create a project, copy files into the project, simulate and view the results. Chapter 1 of the BugHunter Pro and VeriLogger manual has a step-by-step design flow of how to set up a simulator and create a project. You may wish to read that before attempting a to create a complicated project.

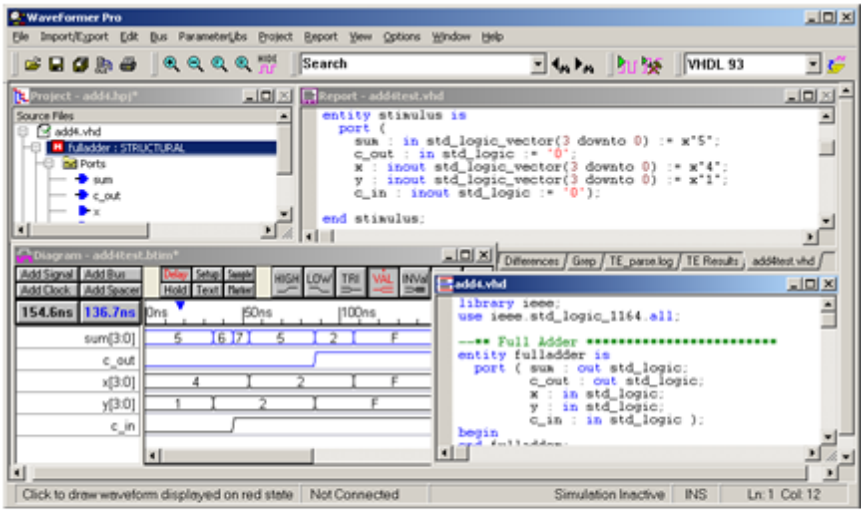


We have also introduced the graphical testbench generation that comes standard with BugHunter Pro. This feature generates a test bench code from a single timing diagram. There are two other levels test bench generation that can be added to BugHunter. The first is the **Reactive Test Bench Option** that generates self-checking code from the expected waveform information. The highest level is **TestBencher Pro** which creates bus functional models from multiple timing diagrams and is able to apply randomized data to each transaction.

Copyright © 2014, SynaptiCAD Sales, Inc.

Simulation 2: Using WaveFormer with ModelSim VHDL

WaveFormer Pro can be used to create a VHDL stimulus file for a VHDL model that needs to be tested. Then the test bench and the model under test can be simulated using an external VHDL simulator. In this tutorial we show the commands to use ModelSim, but if you are using a different simulator this should give you the basic idea for controlling the simulation. Then we will use WaveFormer Pro to compare the simulation results against expected results drawn by the user.



Draw a VHDL test bench and simulate with an external VHDL simulator

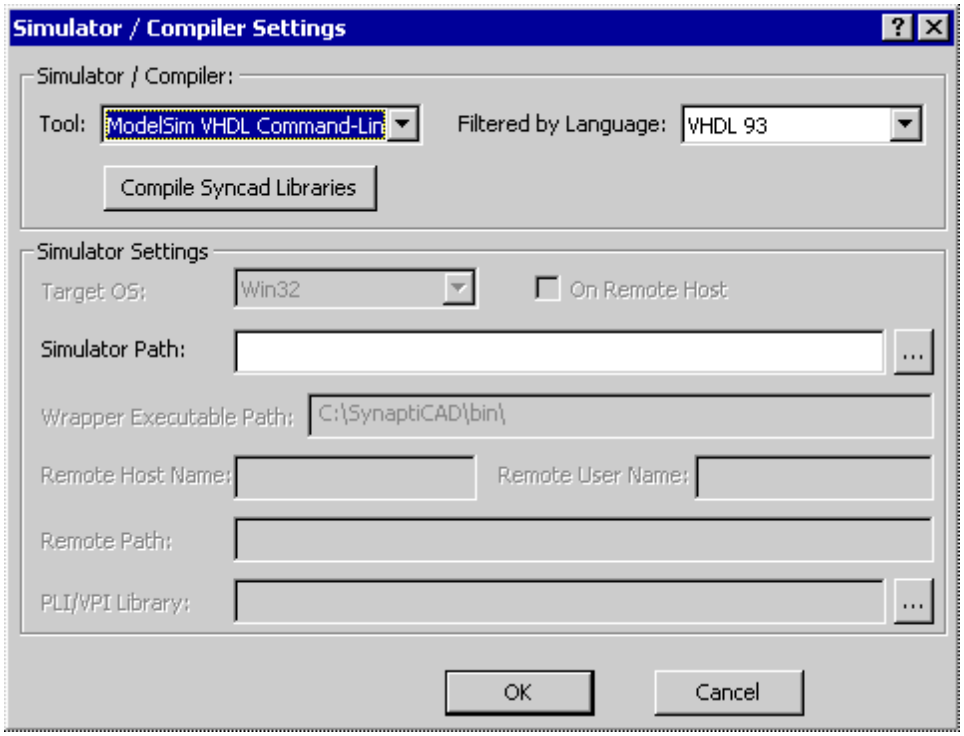
To perform this tutorial, you will need a license for WaveFormer Pro with the Comparison feature and a license for some version of ModelSim VHDL (XE, PE or SE).

(Sim) 2.1 Compile SynaptiCAD Library Models

First time only step: When you first begin using WaveFormer Pro, you will need to compile SynaptiCAD's testbench library models with ModelSim.

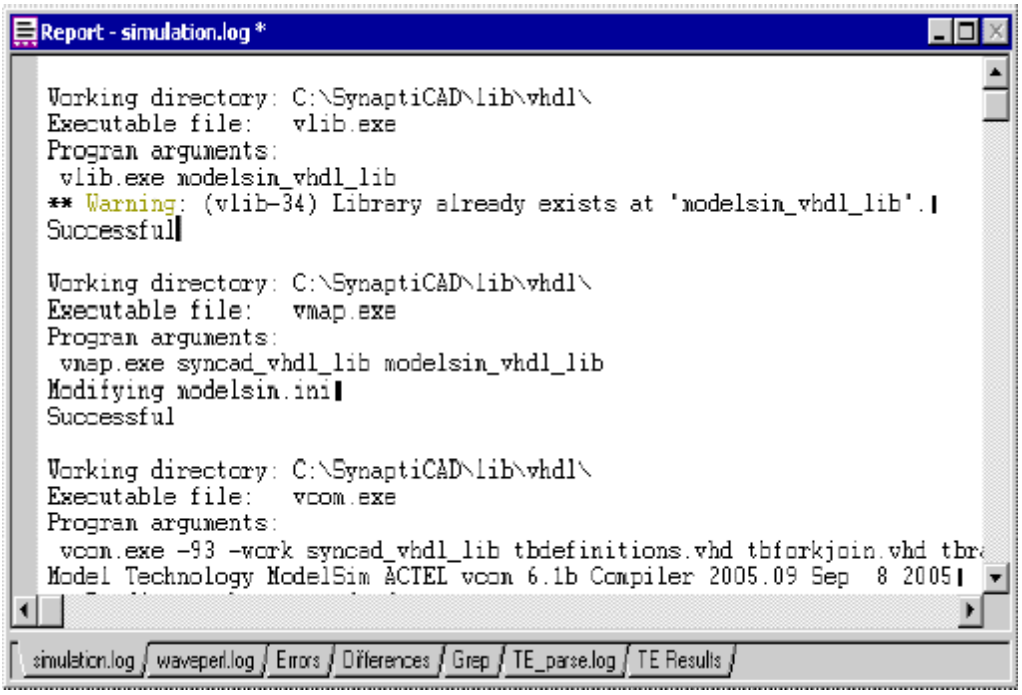
- Start WaveFormer and select the **Options > Simulator/Compiler Settings** menu option to open the *Simulator and Compiler Settings* dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.



- 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**.
- 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.

Copyright © 2014, SynaptiCAD Sales, Inc.

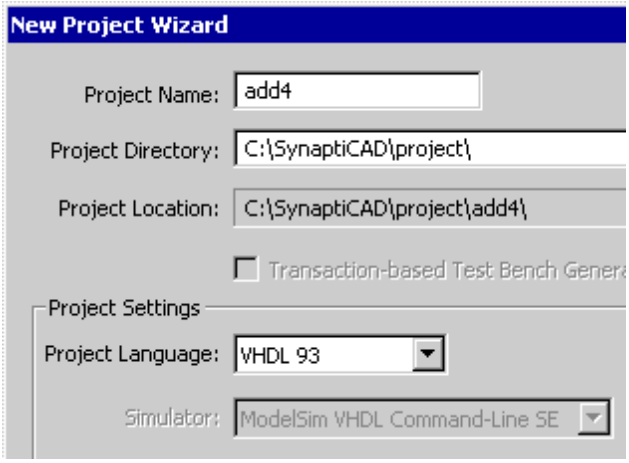


(Sim) 2.2 Create a project and extract the ports

Create a project in WaveFormer Pro to hold the model under test. This will give WaveFormer access to the port information of the model.

In WaveFormer Pro create a new Project:

- Select the **Project > New Project** menu to open the *New Project Wizard* dialog.
- Name the project **add4**. This will be both the name of the project and the directory name where the project is stored.
- Set the **Project Language** drop-down to **VHDL 93**. This tells WaveFormer's parser what language the MUT *(Model Under Test)* is written in.
- Close the dialog to create the project.

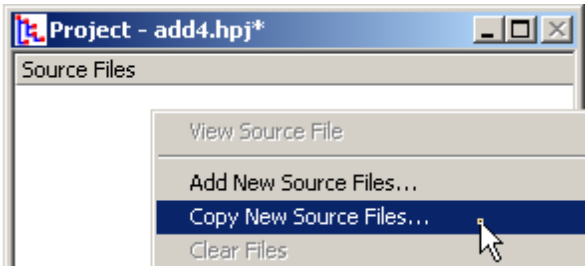


Copy the Model Under Test File into the project:

Next we will copy the file containing the VHDL model under test to the project directory and add the source file to the project file.

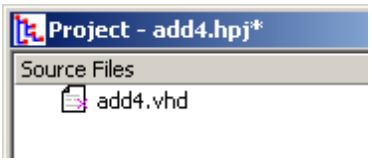
Copyright © 2014, SynaptiCAD Sales, Inc.

- Right click on the surface of the Project window and choose **Copy New Source Files** from the context menu. This will open a file dialog.



- In the above picture, the **copy** command copies the files to the project directory. The **add** command will leave the file in place and just point to the file. For this tutorial we will copy the files.

- In the file dialog, browse to the directory **C:\SynaptiCAD\Examples** and select the **add4.vhd** file. Then press ok to close the dialog. The file should appear in the Project window.

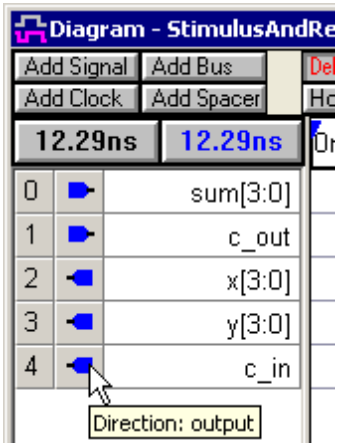


Extract the port information from the Model Under Test:

- Press the **Extract MUT Ports** button on the toolbar.

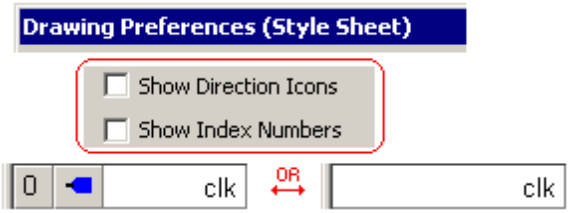


- Notice that the ports of the fulladder model have been inserted into the **Stimulus and Results** diagram. If no signals appear, make sure that language drop-down on the toolbar is set to VHDL.
- Select the **File>Save Timing Diagram As** menu to open a file dialog. Save the diagram and name it **add4test.btim**.



(Optional) Hide the Direction and Index Columns in the Label window:

- By default the Direction and Index columns are shown, but we have hidden them in this tutorial to make smaller images. Choose **Options > Drawing Preferences** to open the dialog. Then uncheck **Show Direction Icons** and **Show Index**.



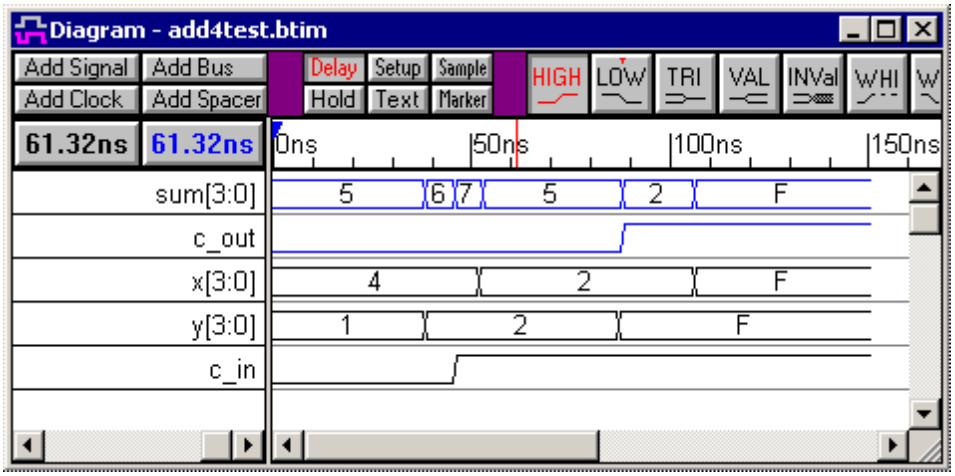
Copyright © 2014, SynaptiCAD Sales, Inc.

(Sim) 2.3 Draw the test bench waveforms

Draw the stimulus on the input ports of the of the MUT and the expected output on the output signals. You can draw anything and see the results, however the rest of the images shown in this tutorial will be using the timing diagram shown below. If you do not want to draw the testbench then you can copy the **add4test.btim** file from **c:\Synapticad\Examples\TutorialFiles\Waveformer2MsimVHDL** directory into the project directory.

Draw the Test Bench

- Draw the following timing diagram. If you have never used SynaptiCAD's drawing environment, then read the rest of the instructions in this section and practice drawing random signals before drawing the following diagram:



- Notice that the ***black*** waveforms are the inputs and will generate code that will drive the MUT. The ***blue*** waveforms are outputs of the MUT and represent expected outputs of the model.
- Save the changes to **add4test.btim** using the **File>Save Timing Diagram** menu, as we will use this btim file later to compare our expected results against the VCD results file generated during simulation.

Basic Drawing Instructions (Skip to Section 2.4 if you can draw the timing diagram)

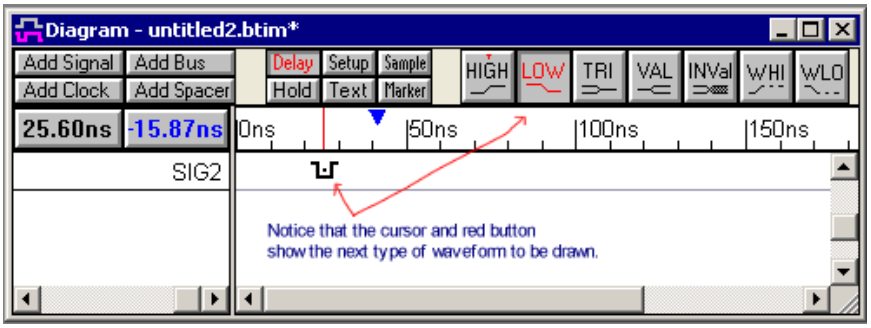
The timing diagram editor is always in drawing mode, so left clicking on a signal will draw a waveform. The red state button controls the type of waveform that is drawn (high, low, tri-state, valid, invalid, weak high, and weak low). The buttons toggle back and forth between two states, and the next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



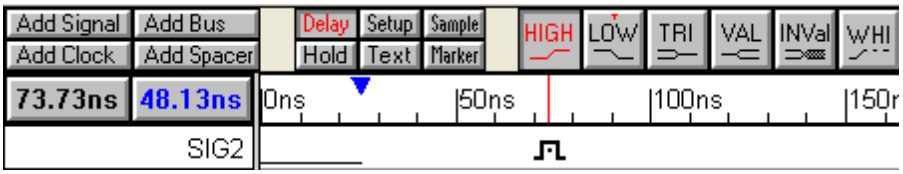
To draw the waveform of a signal:

- 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

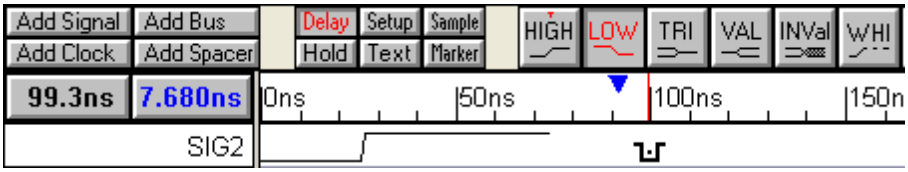
Copyright © 2014, SynaptiCAD Sales, Inc.



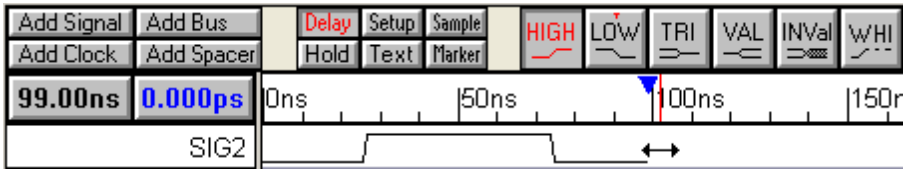
- Click the left mouse button. This draws a waveform from the end of the signal to the mouse cursor.



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



- Keep drawing from left to right across the diagram.

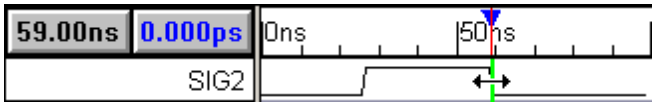


- Pressing the **middle mouse button** either toggles the state buttons or cycles through them depending on the setting in the *Design Preferences* dialog. Choose **Options > Design Preferences** menu to open the dialog.

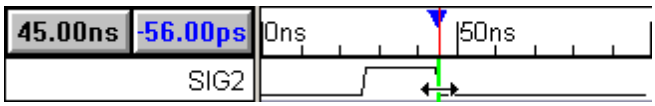
There are several mouse-based editing techniques used to modify existing waveforms. These techniques will only work on signals that are drawn. They will not work on generated signals like clocks and simulated (purple) signals.

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.



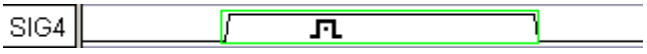
Copyright © 2014, SynaptiCAD Sales, Inc.

- To move transitions on different signals, first select the transitions by holding the **<CTRL>** while clicking on them. Then drag the transition to desired location.



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

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

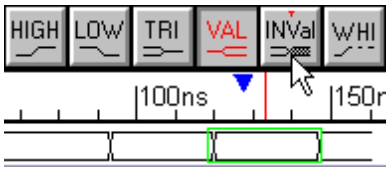


- The inserted state is determined by the red state button

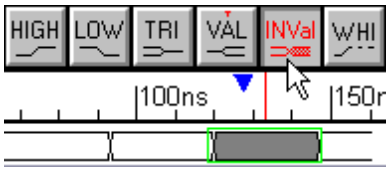


3) Change a segment's graphical state by selecting it and then pressing a state button:

- 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

- 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.

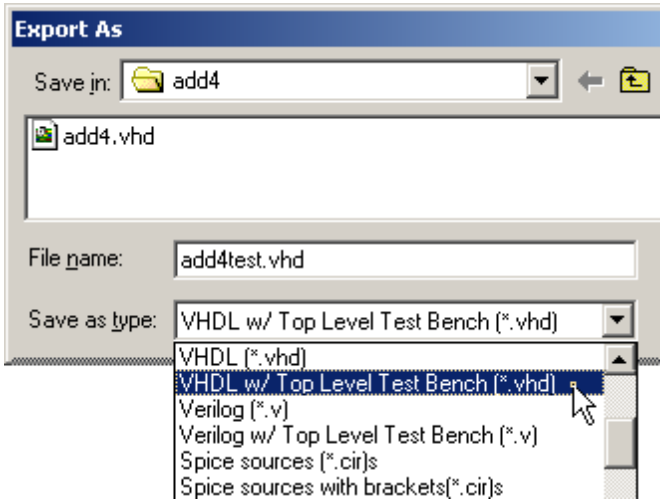
(Sim) 2.4 Export Waveforms to VHDL

In the first steps of this tutorial, we created a project file and pressed the **Extract MUT Ports** button. This also determined the model under test module. The project must remain open during the export of the btim file in order for the model under test to be instantiated in the stimulus file. If the source files in the project contained multiple modules that could be the top level MUT instance, you would need to select the top level instance in the project window by right clicking on the desired top level instance.

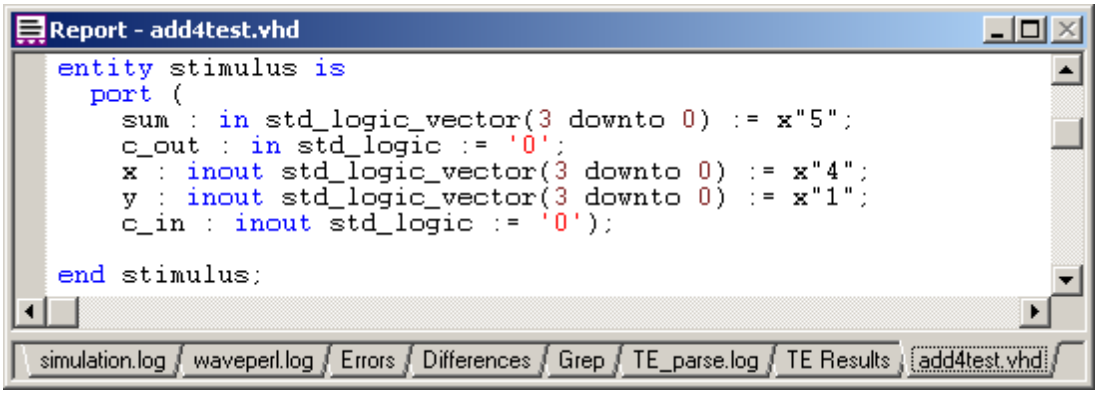
Copyright © 2014, SynaptiCAD Sales, Inc.

Generate the VHDL test bench by exporting the timing diagram file.

- Choose the **Import/Export > Export Timing Diagram As** menu option to open the *Export As* dialog.
- Set the **Save As Type** drop-down to **VHDL w/Top Level Test Bench (*.vhd)**. This creates a VHDL file called **add4test.vhd** containing the stimulus and instantiates a copy of the model under test.



- Once the file is generated it is also loaded into the *Report* window so that you can see the generated code. If you cannot see the *Report* window, then choose the **Window > Report** menu option.



(Sim) 2.5 Simulate VHDL test bench using ModelSim

First map the standard syncad library, syncad_vhdl_lib, so that ModelSim can locate it in this project (this step requires that you have previously compiled the syncad library as described at the beginning of this tutorial). This step only needs to be done once for a project. Next, run the simulation and generate the vcd results file, you can either use the ModelSim *do script* included with this tutorial or manually type the commands.

Map SynaptiCAD library to ModelSim:

- Launch a DOS command prompt and type the following commands:

```
cd \synapticad\project\add4
vmap syncad_vhdl_lib C:\Synapticad\lib\vhdl\modelsim_vhdl_lib
```

EITHER run the do script to simulate, by typing from the command prompt:

```
copy c:\Synapticad\Examples\TutorialFiles\Waveformer2MsimVHDL\add4test.do .
modelsim -do add4test.do
```

Copyright © 2014, SynaptiCAD Sales, Inc.

OR type the following individual commands below

1. To generate library files:

```
vlib work
```

2. To compile the source files:

```
vcom add4.vhd add4test.vhd
```

3. Select the top level module to simulate (this will launch the ModelSim GUI):

```
vsim testbench
```

4. In the console window of the ModelSim GUI, set the name of the vcd dump file:

```
VSIM> vcd file add4.vcd
```

5. Specify the signals to dump to the vcd file (top level signals in the design):

```
VSIM> vcd add /testbench/*
```

6. Simulate the design+stimulus

```
VSIM> run -all
```

7. Exit the simulator (the vcd file will be created by ModelSim at the end of this step):

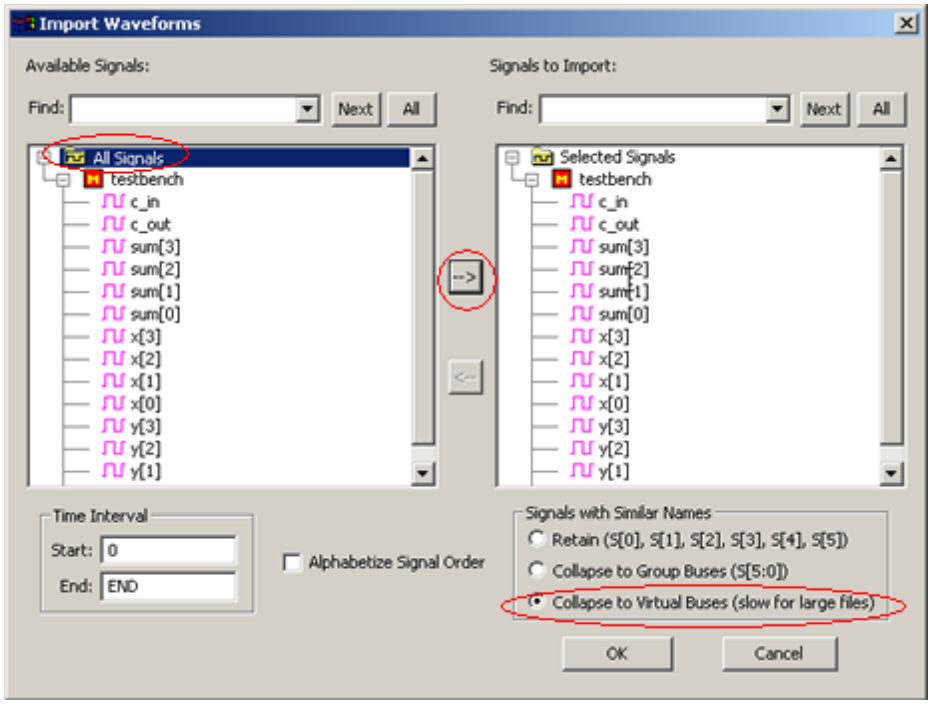
```
VSIM> quit -f
```

(Sim) 2.6 Compare simulation results against expected results

In this step we will compare the simulation results against the expected waveforms that we drew in an earlier section. The compare feature is an option that can be added on to WaveFormer Pro.

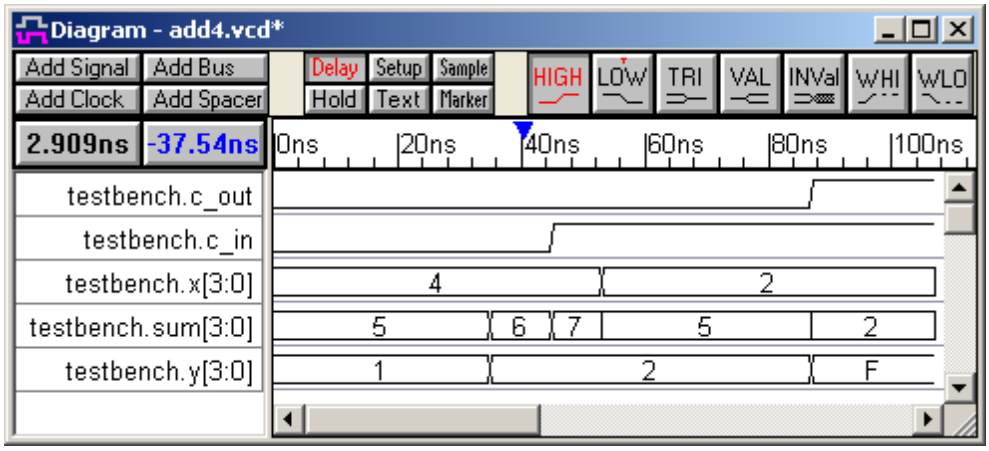
A) Load the Simulation Results file into WaveFormer Pro

- Select the **Import/Export** > **Import Timing Diagram From** menu option and load the **add4.vcd** file created in the previous step to open the *Import Waveforms* dialog.



Copyright © 2014, SynaptiCAD Sales, Inc.

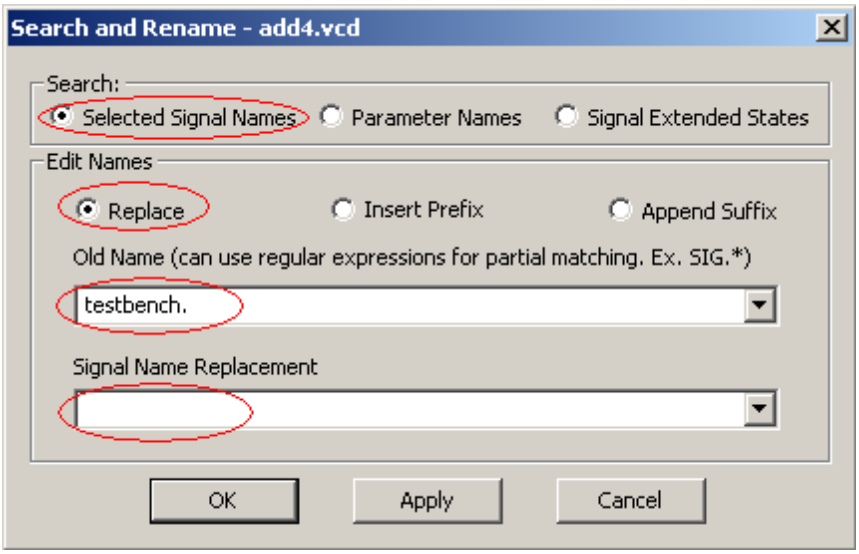
- Check the **Collapse to Virtual Buses** box, so that the signals will be imported as buses instead of as individual bits.
- In the left hand pane, select the **All signals** node then press the => button to move the signals to the right hand pane. Then press **OK** to import the diagram.



B) Strip out the Simulation Model Names

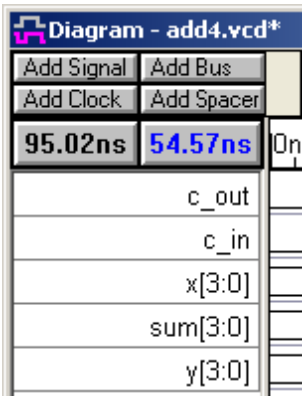
Before we can compare the VCD file to the expected results btim file, we must make sure that the signal names are the same. During simulation, the model name **"testbench."** prefix was added to each of the VCD's signals. We can either strip out the name from the simulation file or add the prefix to the expected waveforms. Either way, we can use the *Search and Rename Signals* dialog to strip or add a prefix. Here we will strip the prefix from the simulation vcd file.

- Select the **Edit > Search and Rename Signals** menu to open the *Search and Rename Signals* dialog.



Copyright © 2014, SynaptiCAD Sales, Inc.

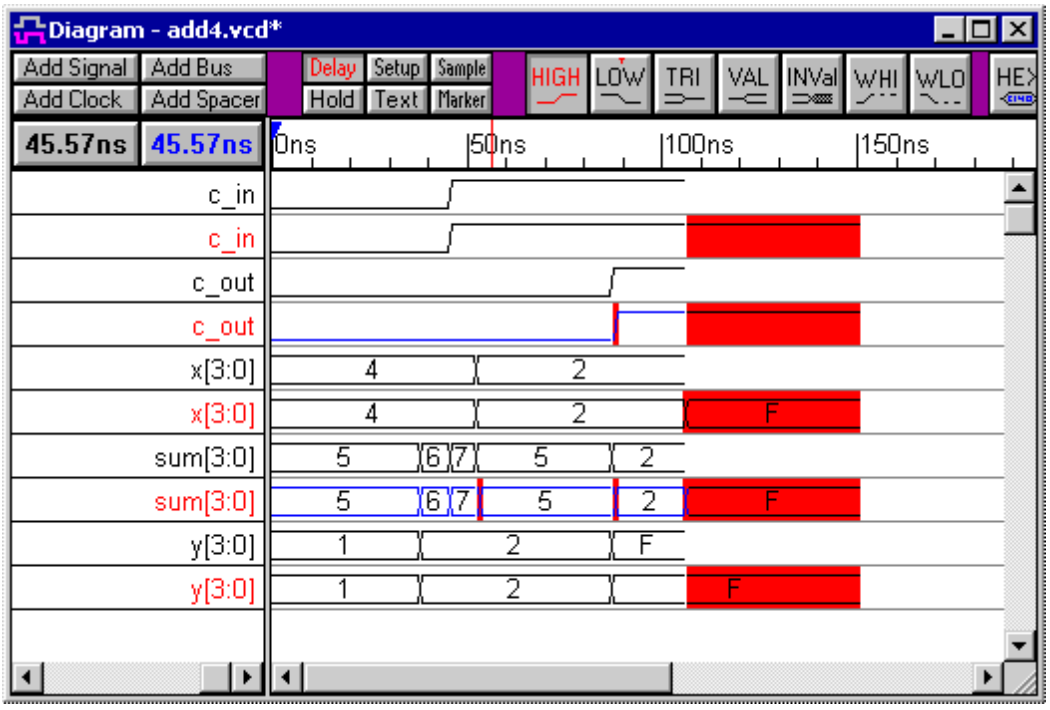
- Check **Selected Signal Names** to make the dialog operate on the signals. Since we did not have any signals selected in the diagram, this will operate on all the signals by default.
- Check **Replace** to make the dialog do a replace. If we had decided to add **"testbench."** to the names in the timing diagram, we would have to use *insert prefix* to add the prefix.
- Set the **Old Name** pattern to **"testbench."** making sure that you include the period after testbench.
- Leave **Signal Name Replacement** blank, because we just want to strip out the name.
- Press **OK** to strip the prefix from the signal names.



C) Compare The Timing Diagrams

To do this step, you will need to have a license for the Compare Feature. This is normally turned on in the evaluation version, but must be purchased in the full version.

- Choose the **File > Compare Timing Diagram** menu to open a file dialog and select the file **add4test.btim** that you created earlier in the tutorial.
- The differences between the VCD file and the expected results file will show up in red. The differences are also displayed in list form in the **Differences** tab in the *Report window.*



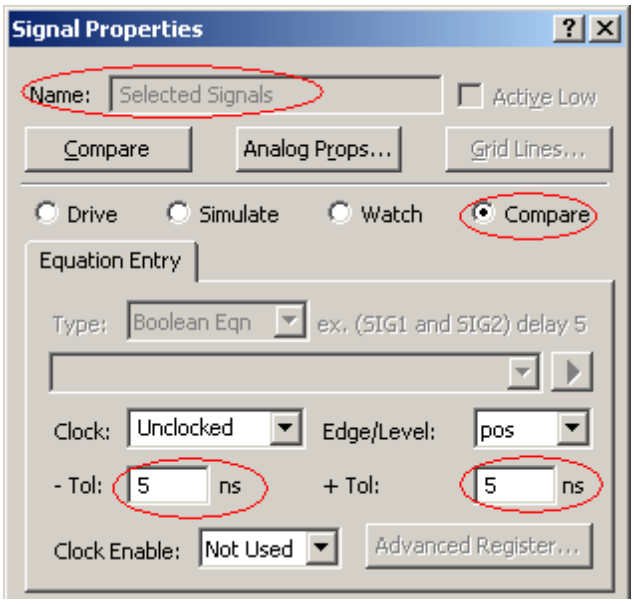
- When drawing the btim file waveforms, you may have some differences with the diagram shown on the tutorial. These differences will show up as highlighted lines in red in the comparison diagram. Minor differences can be removed by using the compare tolerances.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Click the **SET ALL** button on the compare toolbar to open the *Signal Properties* dialog



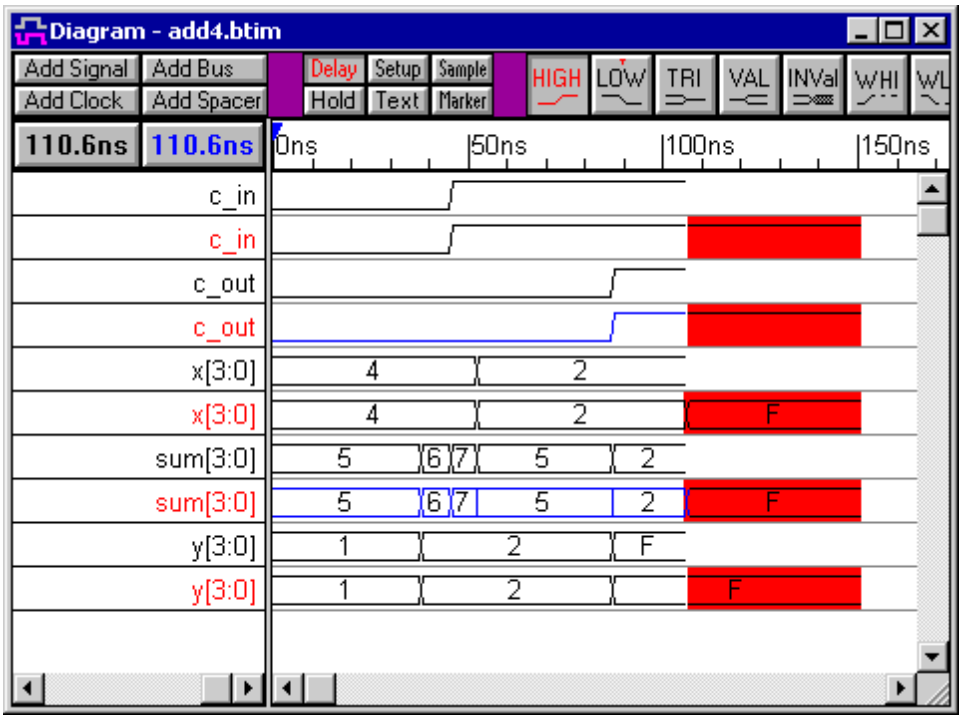
- Notice that the **Name** box is greyed out. This means that the dialog is operating on all the compare signals.
- Type in a tolerance of 5 for both the the **–Tol** and **+Tol** controls. This will allow the compare feature to ignore small changes in values.
- Click the **OK** button to apply the changes



- Press the **Compare All Compare Signals** button to re-compare the waveform signals.

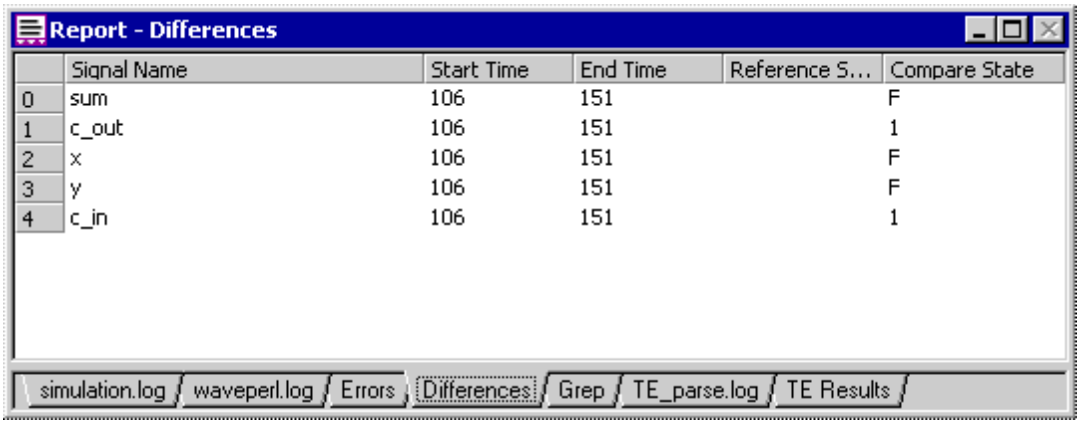


- Below is the resulting diagram. Notice that the thinner lines in red have now disappeared. The remaining differences occur at the end of the waveforms because the simulator stops the simulation as soon as there are no more changes on the waveforms.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Also look at the **Differences** tab in *Report* window, which shows a hyperlinked list of the differences. If you cannot see the *Report* window, then choose the **Window > Report** menu option to bring it to the front.



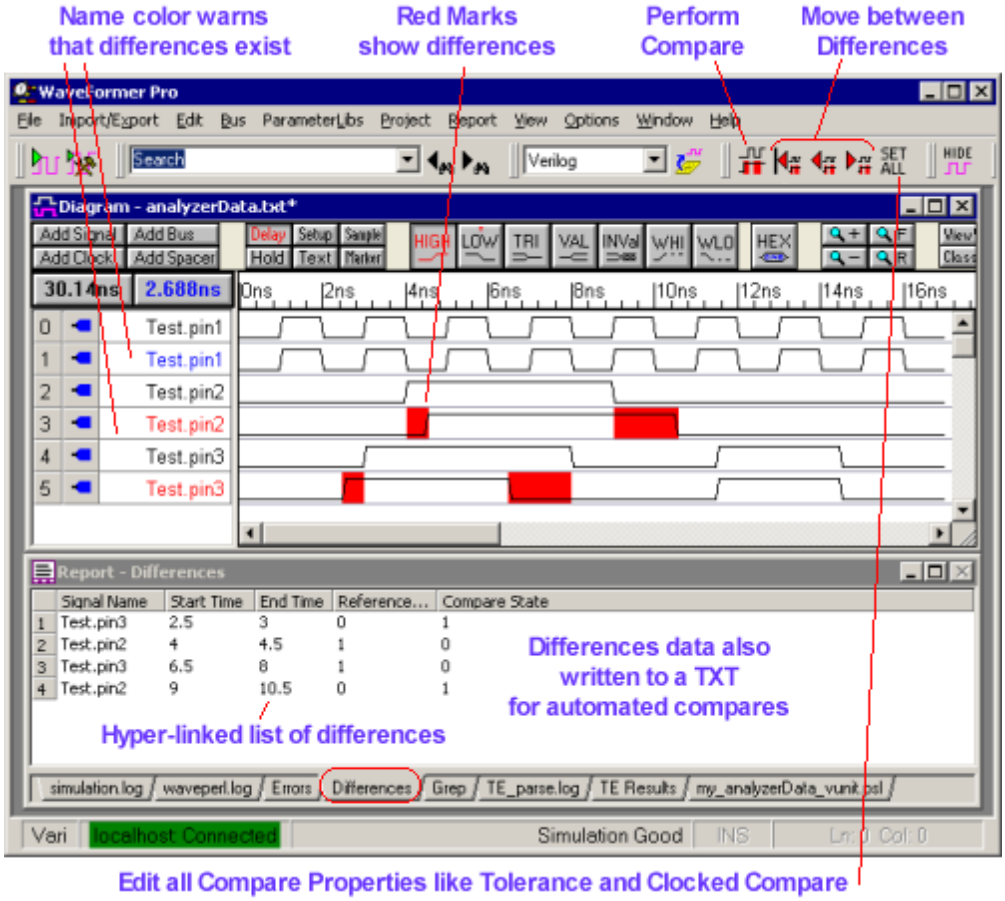
(Sim) 2.7 Summary of Using WaveFormer with ModelSim VHDL

Congratulations! You have now learned how to use WaveFormer Pro with ModelTech's ModelSim VHDL. In this tutorial we covered how to create a new .hpj project in WaveFormer, adding new source files to the project, and drawing the necessary stimulus on the input and output signals. We also learned how to export waveforms to VHDL and how to generate the .vcd file required for testing through ModelSim. Last, but not least, we learned how to compare the simulation results against the expected results.

Copyright © 2014, SynaptiCAD Sales, Inc.

Waveform Comparison Tutorial

Waveform Comparison is an optional module that can be added to most of SynaptiCAD's products that have a waveform editing window. This feature allows comparison between two timing diagrams or between individual signals in a timing diagram. The results of two simulation runs, or of logic analyzer data and a simulation run, can be compared very easily using this feature.



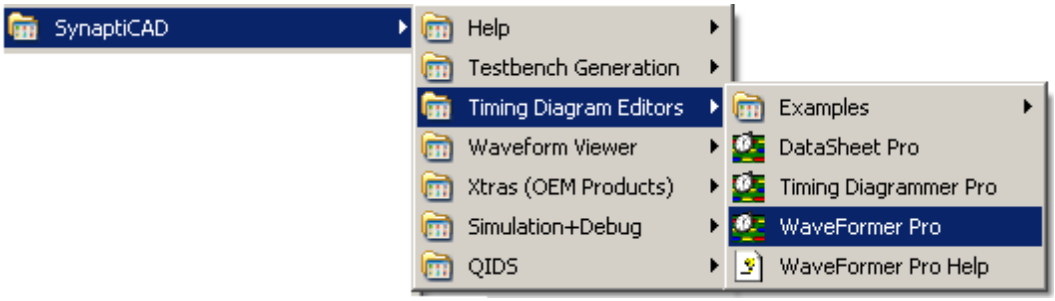
(Compare) 1: Setup for using Compare

This tutorial requires a Compare Option license plus a license for one of the SynaptiCAD products that supports this feature. To obtain a temporary license for evaluation purposes, complete the form under the **Help > Request License** menu item and contact our sales department.

Run WaveFormer Pro or higher:

- Run WaveFormer Pro, DataSheet Pro, VeriLogger, or one of the more advanced products. If you are evaluating Timing Diagrammer Pro and you would like to learn about the compare features, close the program and restart the evaluation version in WaveFormer Pro mode.

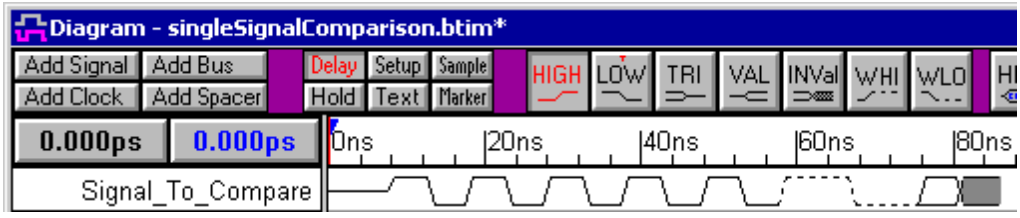
Copyright © 2014, SynaptiCAD Sales, Inc.



Load the Starting Timing Diagram:

This tutorial uses several files contained in the the **Examples\TutorialFiles\WaveFormComparison** subdirectory of the installation directory (**C:\SynaptiCAD** by default on Windows). These files are examples of simulation results, logic analyzer data, and timing diagram files. To get started:

- Open the file **singleSignalComparison.btim** in the **SynaptiCAD\Examples\TutorialFiles\WaveFormComparison** directory.



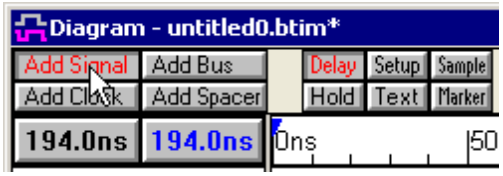
- Select the **File > Save As** menu option, and save this file as **mysingleSignalComparison.btim**.

(Compare) 2: Individual Compare Signals

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.

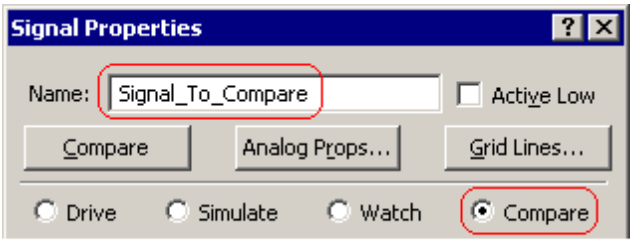
Add a Signal and set its compare setting and change its name:

- Press the **Add Signal** button to add a signal to the diagram.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Double click on the name of the new signal (**SIG0**) to open its *Signal Properties* dialog.
- Change the name to **Signal_To_Compare** to exactly match the other signal in the diagram.
- Set the signal type to **Compare**.
- Notice that the name of the compare signal is blue.



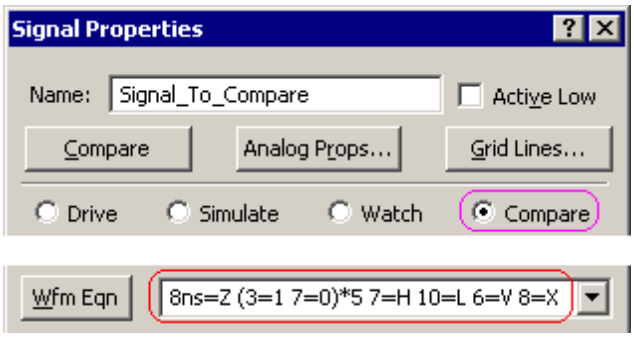
Draw the waveform for the Compare Signal:

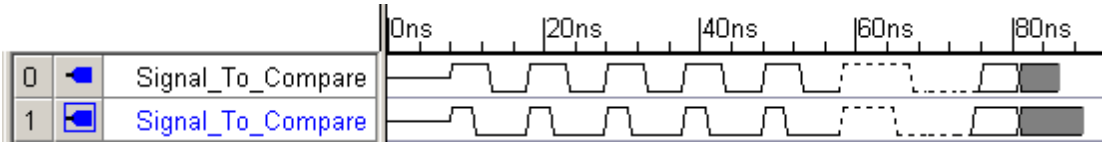
We will use the Waveform Equation generator to create a waveform that is similar to the one on the original signal.

- In the *Signal Properties* dialog of the compare signal, modify the expression in the **Wfm Eqn** so that it reads:

```
8ns=Z (3=1 7=0)*5 7=H 10=L
6=V 8=X
```

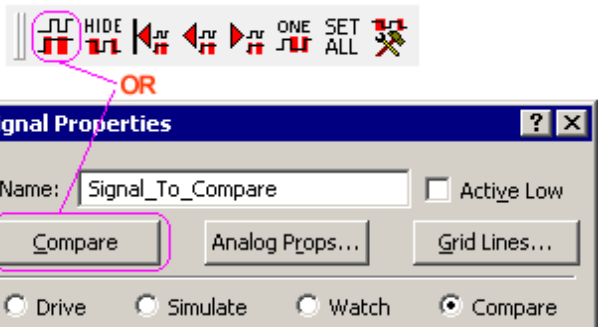
- Press the **Wfm Eqn** button to drawn the specified waveform.

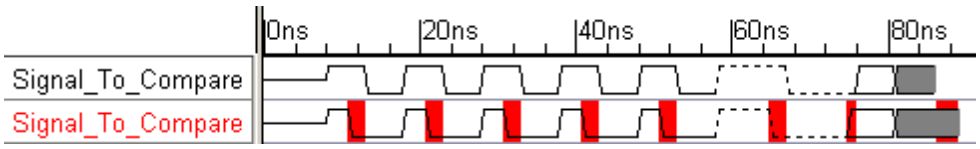




Perform the Compare:

- Press the **Compare** button in either the dialog or on the button bar to calculate the comparison.
- Press the **OK** button to close the *Signal Properties dialog.*
- The red portions of the compare signal mark the areas where differences were detected. Also notice that the name is in red to indicate that there are differences somewhere on the signal.





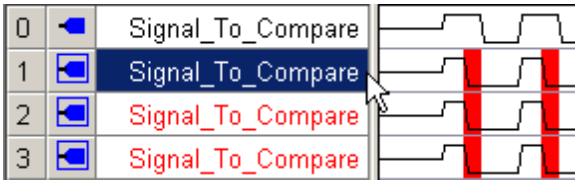
Copyright © 2014, SynaptiCAD Sales, Inc.

(Compare) 3: Experiment with Tolerance

Each Compare signal has a plus and minus tolerance setting. The tolerance settings specify regions around the reference signal in which to ignore any differences. This is one method for ignoring small differences during a comparison. In this section we will copy the compare signal a couple of times and set each compare signal with a different tolerance.

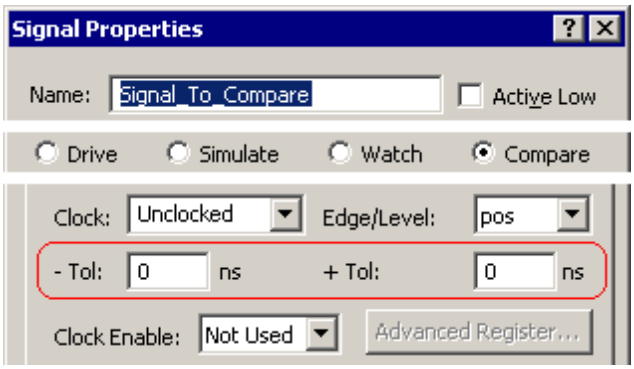
Copy the Compare signal and paste it twice:

- Select the Compare signal by clicking on the red **Signal_To_Compare** name.
- Press **<Ctrl>C** keys on the keyboard to copy the signal.
- Select the **Edit > Paste Signal(s)** menu option or press <**Ctrl**>**P** to paste a copy of the original compare signal.
- Past again to create a third compare signal.



Set the Tolerance settings:

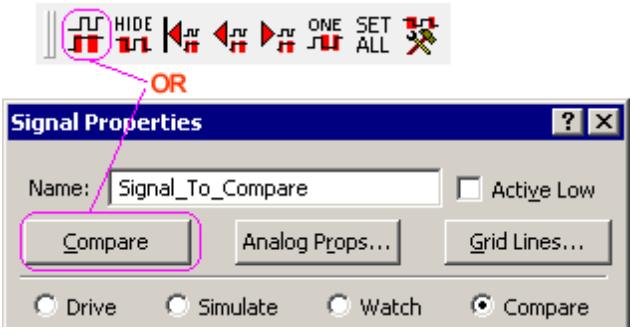
- Double click on first compare signal to open the *Signal Properties* dialog.
- Notice that the tolerance settings are at zero. This means that any variation will be flagged as an error.
- We will leave this signal at zero tolerance so that you can compare it to the other two signals



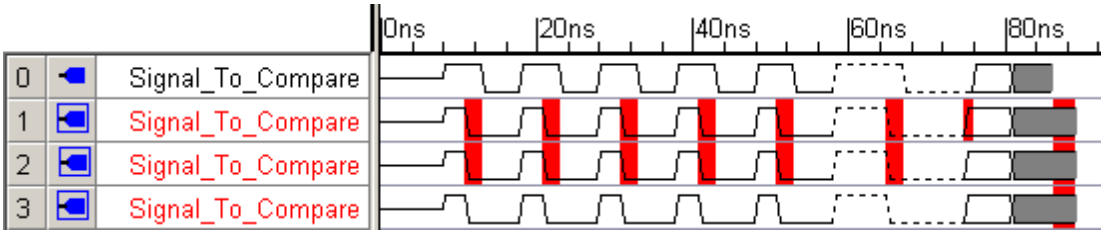
- Double click on the second compare signal and change its **+Tol** to **2**.
- Double click on the third compare signal and change its **+Tol** to **3**.



- Notice that the waveforms are still displaying the original compare information. This is because compares are only calculated when you press the compare button.
- Press the **Compare** button to perform the compare.

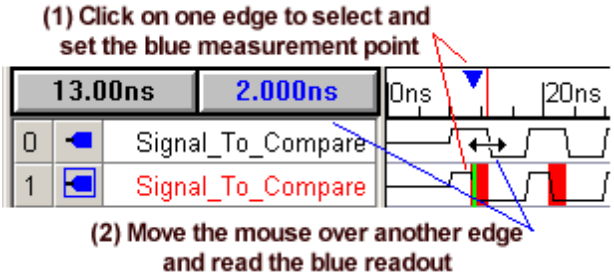


Copyright © 2014, SynaptiCAD Sales, Inc.



- Notice that the second compare signal only has one less difference than the first.
- Notice that the third compare signals with a tolerance of 3 has eliminated most of the differences between itself and the original signal.

- You can experiment with the Tolerance setting by dragging edges on the compare signals and the pressing the compare button.
- Measure the differences in the edges by using the mouse button and the Blue readout.

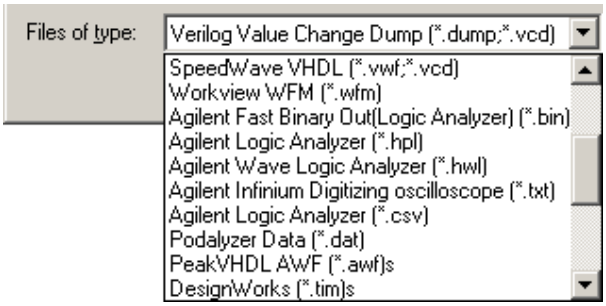


(Compare) 4: Compare Timing Diagrams

When comparing two files, the file that is being compared/merged into the reference diagram must be a SynaptiCAD btim file. However, since WaveFormer reads many different file formats, it is a simple operation to translate a waveform file to .btim. In this step we will first convert a VCD (Verilog simulation file) into a .btim file. Then we will use the converted file to compare against a logic analyzer file. This shows how a simulation file could be compared with data captured from an actual circuit.

Import the VCD file and save it as a BTIM file:

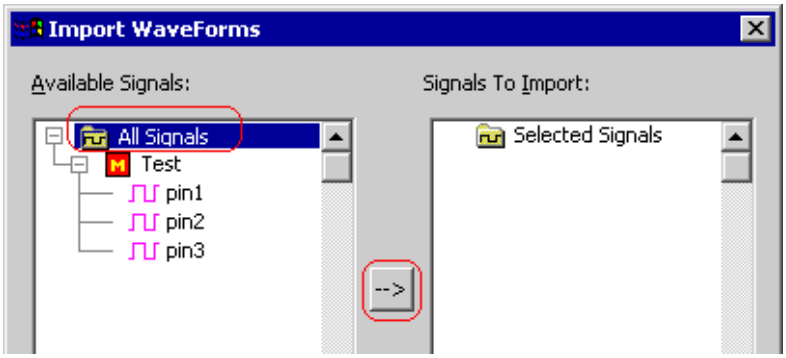
- Select the **Import/Export > Import Timing Diagram From** menu to open a special version of the *Open Timing Diagram* dialog that remembers the file type of the last file imported.
- Select the **Verilog Value Change Dump (*.dump, *.vcd)** option from the *File of Type* list. This is a file generated by a Verilog simulator.



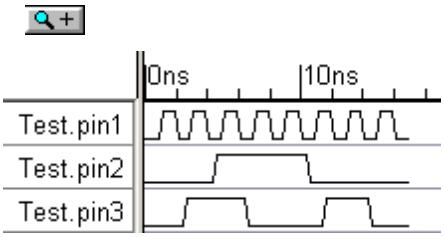
- Select the **simulationResults.vcd** in the **SynaptiCAD\Examples\TutorialFiles\WaveFormComparison** directory.
- Press **Open** to close the file dialog and launch the *Import Waveforms* dialog. This dialog allows you to selectively load signals from really large files.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **All Signals** folder in the *Available Signals* side of the dialog by left clicking on it.
- Press the right arrow button to place these signals in the *Signals to Import* tree.
- Press **OK** to close the dialog and import the signals.



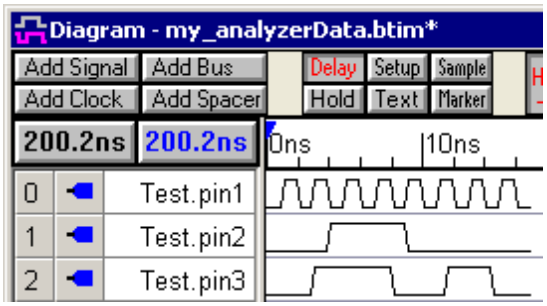
- Use the **Zoom In** button to get a better view of the changing signal edges.
- Choose the **File > Save Timing Diagram As** menu to open the *Save File* dialog.
- Choose **Timing Diagram - Binary (*.btim)** from the type drop-down and save the file as **simulationResults.btim**.



Load the Logic Analyzer Data file:

Next we will open a data file that is in a Spreadsheet format similar to that generated by a Tektronix logic analyzer.

- Select the **File > Open Timing Diagram From** menu to open the *File* dialog. We are opening the file this way just to skip the *Import Waveforms* dialog step that was demonstrated with the VCD file load.
- Select the **Test Vector Spreadsheet/Tektronix (*.txt)** option from the *File of Type* list.
- Select the **analyzerData.txt** in the **SynaptiCAD\Examples\TutorialFiles\WaveFormComparison** directory.
- Use **File > Save Timing Diagram as** menu to save the file as **my_analyzerData.txt**

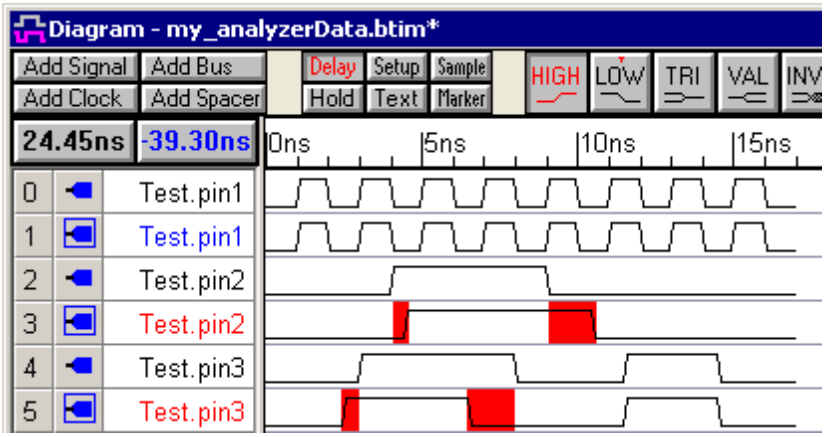


Compare the diagrams:

When two diagrams are compared, the signals from the second diagram selected are brought into the first diagram as compare signals. These signals can either be grouped at the bottom of the open diagram or the Compare signals can be 'interleaved' with the original signals. That is, the Compare signal for a given original signal will be inserted directly under the original signal. This behavior is controlled using the **View > Compare and Merge > Interleave Compare and Merge Signals** menu option. This tutorial will have the interleave feature turned on.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **File > Compare Timing Diagram...** menu option to launch the *Compare* dialog. Notice that the default file type is **Timing Diagram (*.tim, *.btim)**.
- Select the **simulationResults.btim** file that you converted earlier in this section.
- Click **Open** to compare the two diagrams.



- Notice that the Compare signals are placed immediately following the signals that they are being compared to since the **Interleave Compare and Merge Signals** option is on.
- Notice that two of the signal labels, **Test.pin2** and **Test.pin3**, have turned red because there are differences on these signals.

Tip: If the data sets being compared have slightly different naming schemes, then the signals will not properly interleave because the program will not be able to properly match the signals for comparison. The **Edit > Search and Rename Signals** feature is handy for modifying one of the sets of signal names in the case. This feature performs pattern matching on the signal names and allows you to replace characters in the name, or append/remove a prefix or a suffix to the signal name. We will demonstrate this in Section 9 of this tutorial.

(Compare) 5: Set All Compare Signal Properties

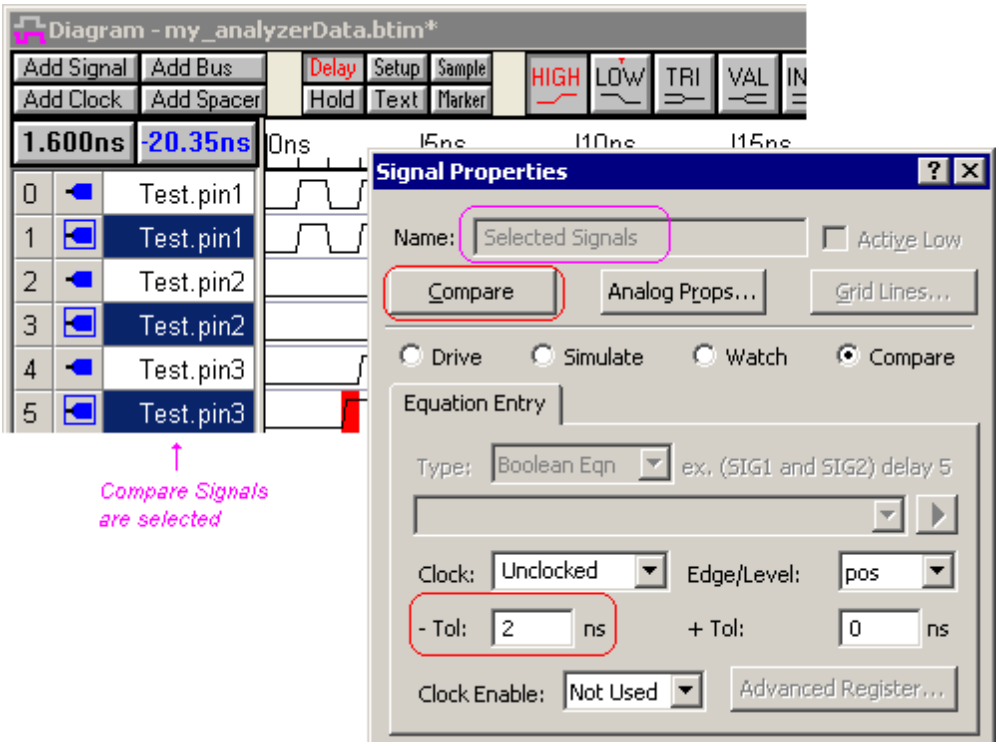
In section 3 we showed how to edit the properties of one compare signal by double clicking on the signal's name and making the changes in the *Signal Properties* dialog. However, it is typically desirable to edit all of the compare signal properties at the same time. The SET ALL button on the Compare button bar will select all of the compare signals and open the *Signal Properties* dialog in a special mode for editing all of the selected signals.

- Either press the **SET ALL** button or choose the **View > Compare and Merge > Edit Compare Signals** menu to select all the compare signals and open the *Signal Properties* dialog.

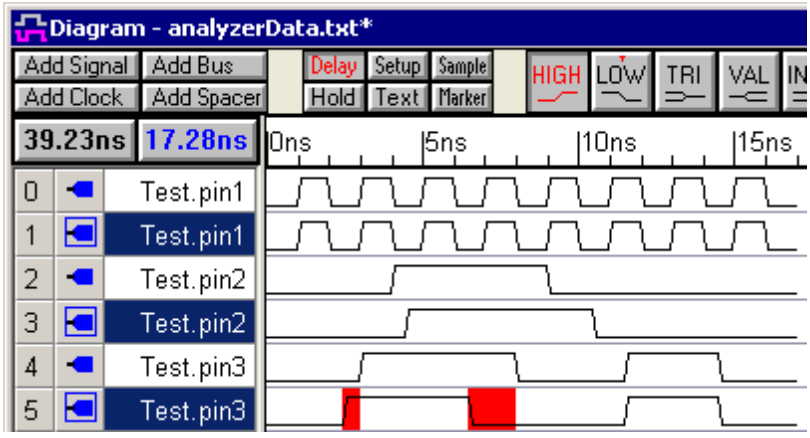


- Note: You can choose to deselect individual signals prior to modifying the signal properties. This may be useful if you want to specify a Tolerance range for all but one signal, for example.

Copyright © 2014, SynaptiCAD Sales, Inc.

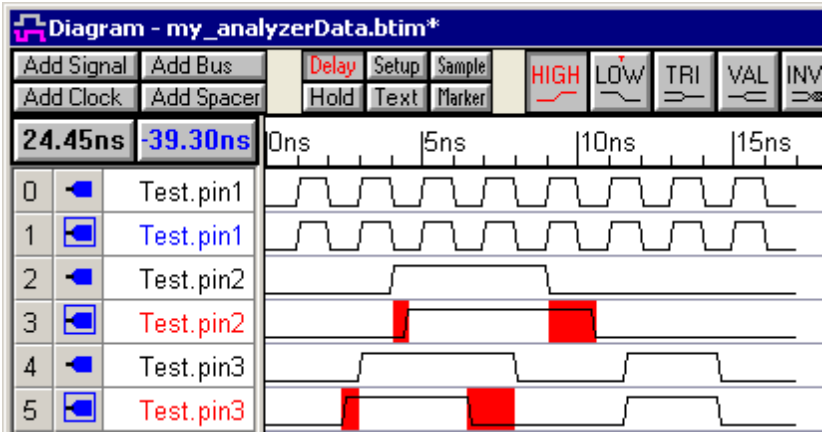


- Notice that only the compare signals are selected.
- Also notice that the **Name** box in the *Signal Properties* dialog is grayed out to show that you are editing only properties that can be the same between the selected signals.
- Set the **-Tol** to 2 ns.
- Press the **Compare** button.



- Notice that with the -Tolerance set at **2ns**, the **Test.pin2** signal no longer has any differences to report.
- Set the *-Tolerance* back to **0ns** and press the **Compare** button, so that we will have more differences to look at in the next section.
- Close the *Signal Properties* dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.



(Compare) 6: Find the Differences

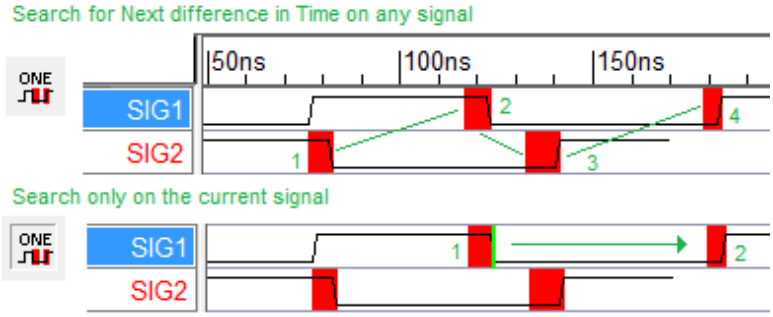
So far we have found the differences between signals by looking for the red marks on the compare signals, which is sufficient for small files. However for very large files you may want to use one of these methods for finding the differences:

(1) Use the Compare tool bar to show the next difference:

- The Arrow buttons on the Compare toolbar are used to move between the differences in the diagram.



- The **One** button determines how the next and previous searches are done. Normally searches find the next difference in time on any signal. If the **One** button is depressed, then the searches will only be done on the current signal.



- Press the **Move Next** button a few times and watch how the selection in the diagram moves to the next difference in time. Then depress the **One** button and press the **Move Next** button a few times and notice that the searches are constrained to a single signal.



- Press the **Move Previous** button and watch how to selection in the diagram moves to the previous difference.



- Use the mouse to select an edge on a compare signal and then use the **Move Next** or **Move Previous** to jump to the next or previous difference.

- Press the **Move to First Difference** and watch how it highlights the first difference found after time 0 in the diagram.



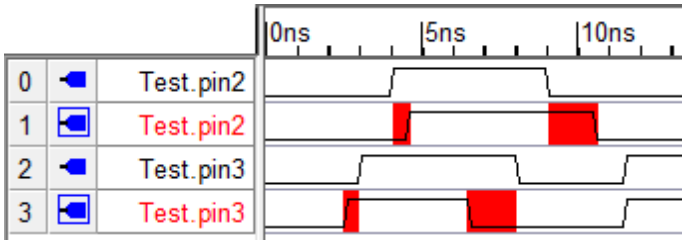
Copyright © 2014, SynaptiCAD Sales, Inc.

(2) Hiding Signals Without Differences (matching signals)

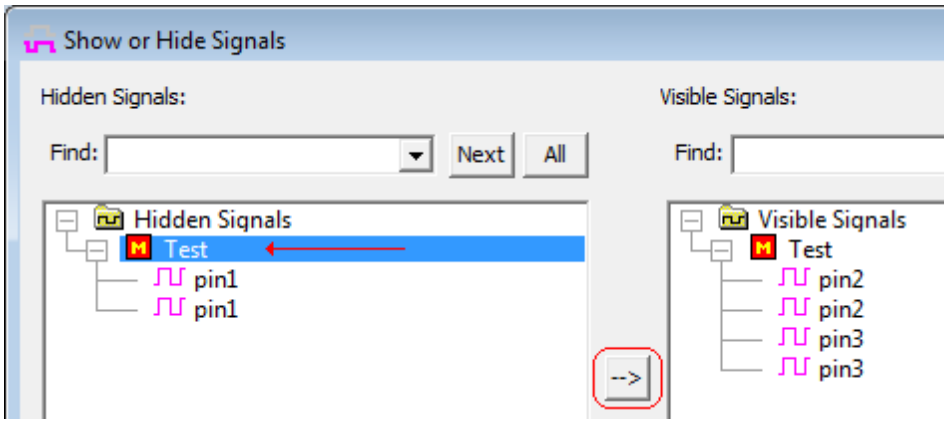
- Press the **Hide** button to hide signal pairs that do not have any waveform differences.



- Notice that both Test.pin1 signals are no longer shown in the waveform display.



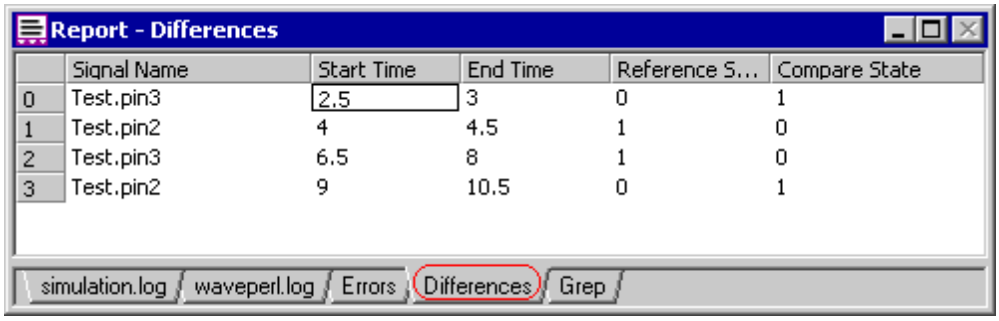
- Once signals are hidden, the **View > Show and Hide Signals** menu can be used to re-show the hidden signals. Select the hidden signals individually or by selecting a parent node (e.g. select the Hidden Signals to select all the hidden signals for un-hiding). Then press the arrow button to move the signals to the ***Visible Signals*** list. When you close the dialog all 6 signals should be visible.



(3) Double click on a line in the Differences Tab of the Report Window

After a compare is performed, there will be a tab in the Report window called Differences that lists one difference per line in time order.

- Arrange the *Report* and *Diagram* window so that you can see both windows at the same time. If your *Report* window is not visible, select the **Windows > Report Window** menu option to bring the window to the foreground.
- Click on the **Differences** tab to bring this page to the front of the *Report* window.



Copyright © 2014, SynaptiCAD Sales, Inc.

- In this example, the first difference in time is on **Test.pin3**. The difference started at time 2.5ns and ended at time 3ns. The Reference Signal had state value of 0 while the Compare Signal had a State of 1.
- Double click a line in the *Report* window and notice how the corresponding difference in the diagram window is also highlighted. If this was a very large file and the difference was not displayed in the window, the diagram would have also scrolled over so that you could see the highlighted difference.

(4) Use an external program to process the Differences File

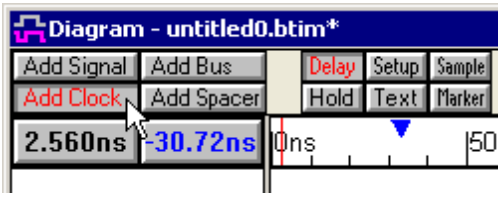
The data displayed in the *Differences* tab of the *Report* window is loaded from a tab-delimited text file generated during the comparison. This data can be backed up and stored for later reference. The file in this example is named **analyzerData_diff.txt** (formed from <referenceFileName>_diff.txt). You can use the *Report* window to view this file by selecting the **Report > Open Report Tab** menu item to open the file.

(Compare) 7: Perform a Clocked Comparison

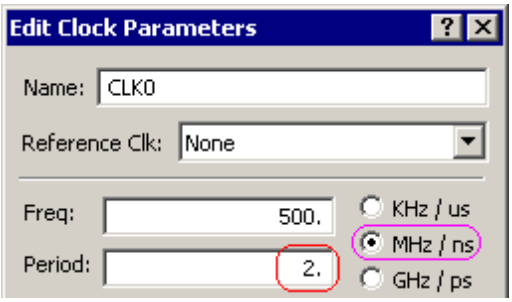
A clocked comparison compares the signal states at clock edges instead of continuously. In this section we will add a clock to the diagram and then set the clock properties of all of the compare signals to reference the clock.

Add a clock signal with grid lines on the positive edges:

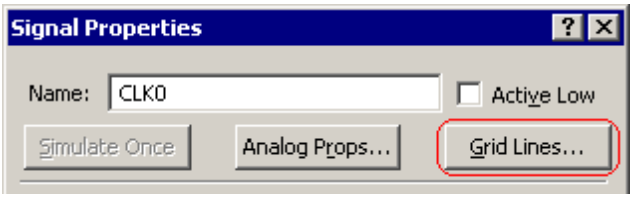
- Press the **Add Clock** button to open the *Edit Clock Parameters* dialog.



- Set the *Period* of the clock to **2ns**.
- Press the **OK** button to close the dialog and add the clock to the diagram.
- Double click on the **CLK0** name to open the *Signals Properties* dialog.

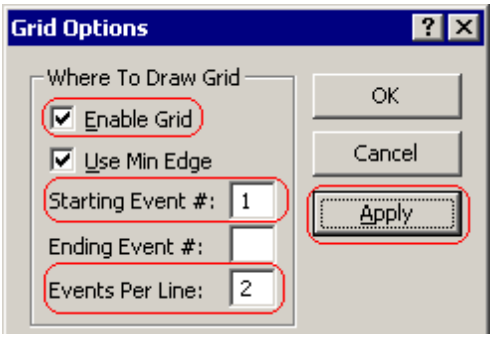


- Press the **Grid Lines** button to open the *Grid Options* dialog. We will add a grid to the positive edge of the dialog so that it will be easy to see the sampling points on the diagram.

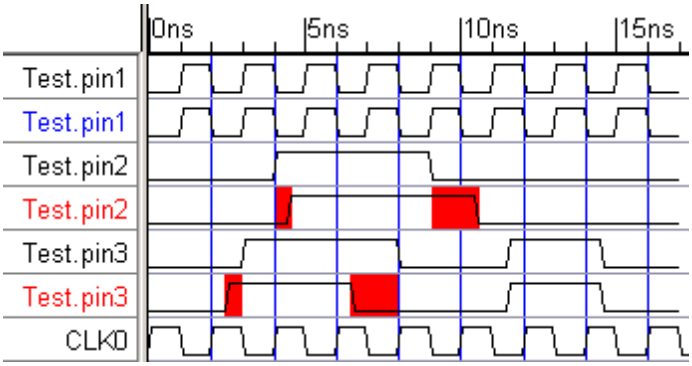


Copyright © 2014, SynaptiCAD Sales, Inc.

- Check the **Enable Grid** box to enable the controls in the dialog.
- Enter a **Starting Event** of **1**. This means that the first grid line is drawn on the first event of the clock (in this case it is at 0ns on the positive edge).
- Enter an **Events Per Line** of **2**, so that the grid only draws on every-other line.



- Press the **Apply** button and make sure that the gridlines are on the positive edges of the clock.
- Press Ok to close the *Grid Options* dialog, then press Ok to close the *Signal Properties* dialog.

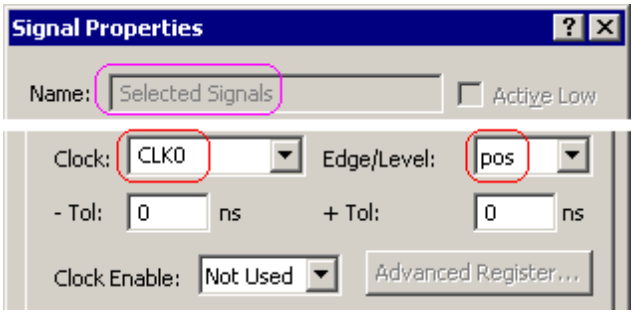


Make the Compare Signals reference the sampling clock:

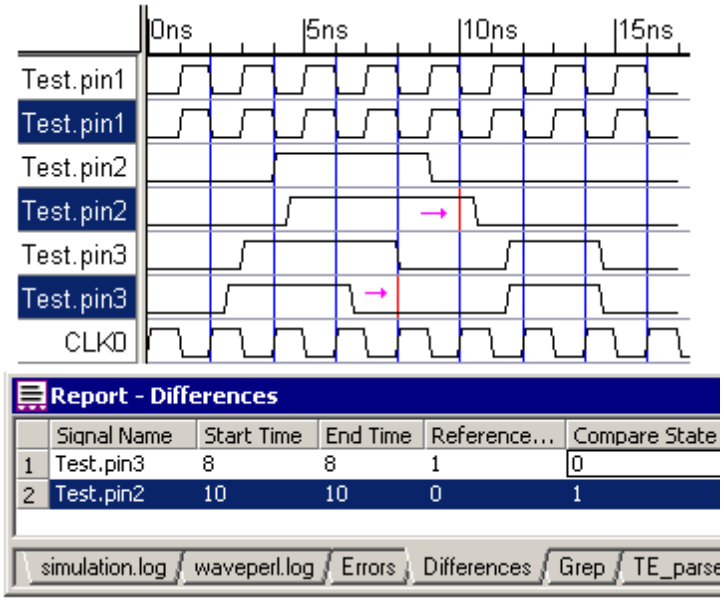
- Either press the **SET ALL** button or choose the **View > Compare and Merge > Edit Compare Signals** menu to select all the compare signals and open the *Signal Properties* dialog.



- Set the *Clock* to **CLK0** and set the *Edge/Level* to **pos**.
- Press the **Compare** button to apply the changes and perform a new comparison.



Copyright © 2014, SynaptiCAD Sales, Inc.



- Notice that there are now only two differences in the diagram. These two differences occur on the **8ns** (on **Test.pin3**) and **10ns** (on **Test.pin2**) rising edges of the clock.
- You can experiment with changing the clocking edge to **neg** to see how the compare changes.

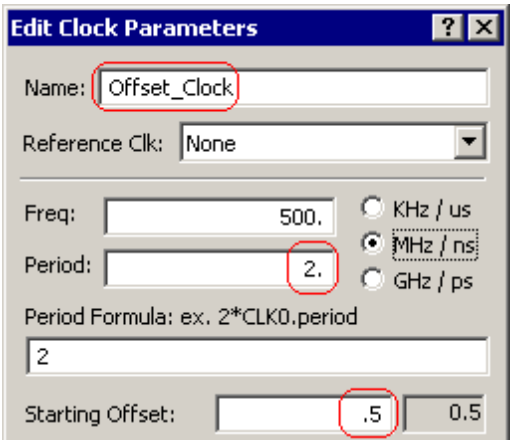
(Compare) 8: Compare During Clock Cycle Windows

The clocked comparison described in the previous step provides for comparison in state of signals around clock edges, but sometimes you need to check for signal differences during a window of time that is relative to the clock cycle, but not around the clock edge itself. You can use a second clock, offset from the first, to create windows during which to compare during the high or low segments of the original clock instead.

Add Offset Clock to Diagram

Since the window comparisons in this example are to be relative to the original clock, the offset clock will have the same frequency and period as the original clock, but have a different starting offset value.

- Add a second clock to the diagram and set the following in the *Edit Clock Properties* dialog.
- Set the *Name* to **Offset_Clock**, the *Period* to **2ns**, and the *Starting Offset* to **.5**.
- Press **OK** to close the *Edit Clock Properties* dialog and apply the changes.
- Notice the **Offset_Clock** now starts halfway through the first segment of **CLK0**.



Copyright © 2014, SynaptiCAD Sales, Inc.

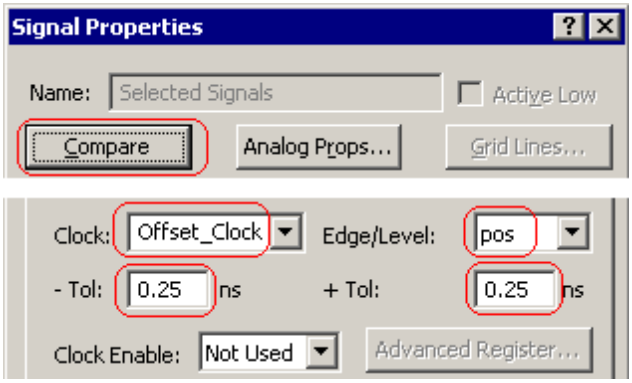
Change the Compare Signals to Use the Offset Clock

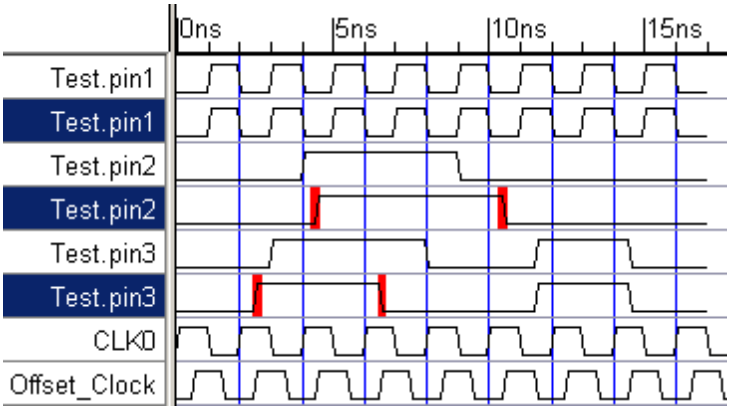
Next, change the three Compare signals to use the **Offset_Clock** as the clocking signal and use the Tolerance settings to create the window for comparison:

- Either press the **SET ALL** button or choose the **View > Compare and Merge > Edit Compare Signals** menu to select all the compare signals and open the *Signal Properties* dialog.

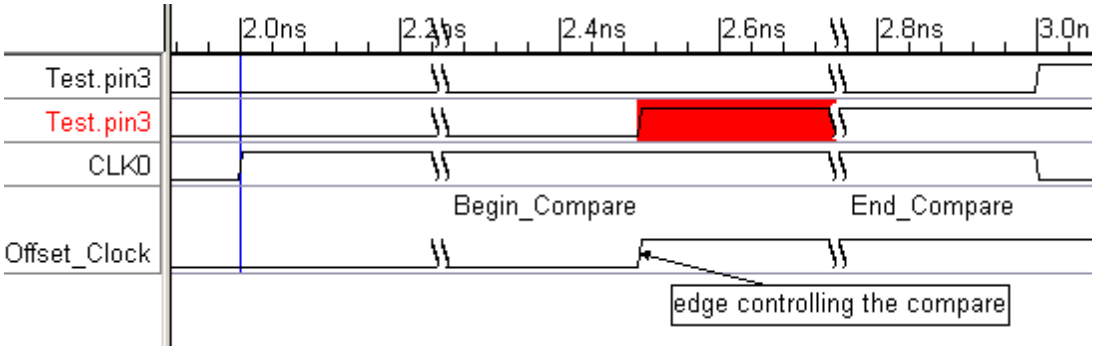


- Set the *Clock* to **Offset_Clock** and the *Edge/Level* to **pos**.
- Set both *Tolerance* values to **0.25** to create the testing window.
- Press the **Compare** button to apply the changes and perform the comparison.





- Below is an image that we made by first zooming into the diagram. Then we placed two markers around the sampling clock edge at exactly -.25 ns and +.25 ns around the edge to show the sampling region created by the Tolerance setting. We also used a text object attached to the clock edge to call attention to it.
- Notice that the difference between the two **Test.pin3** signals continues beyond the compare window.



Copyright © 2014, SynaptiCAD Sales, Inc.

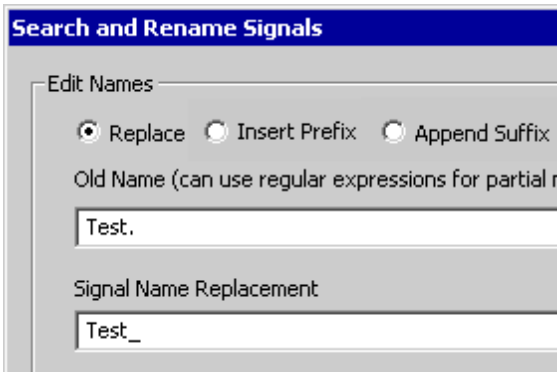
(Compare) 9: Mask Sections to Exclude Comparison

It is sometimes useful to mask some segments during comparison. In this example, we will create a **Compare_Enable** signal that will specify the specific clock segments over which we want the comparison to be performed. Then we will use a Boolean equation (**Offset_Clock and Compare_Enable**) in a simulated signal to define the region in which to perform the compare. Since this step uses simulated signals you will need to be using a product that supports this feature like WaveFormer Pro (not one of the Viewers).

Search and Replace Signal Names:

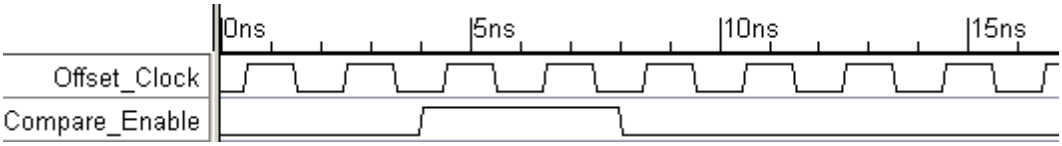
First, we will change the names of the signals in the diagram so that when we add the simulated signal, we do not get simulation errors (with the signal names Test.* the simulator will look for a module that we will not have defined).

- Press **<CTRL>A** to select all of the signals in the diagram. The Search and Rename signals feature will search the selected signals. If no signals are selected, then a dialog will appear asking if you would like to select all signals in the diagram.
- Select the **Edit > Search and Rename Signals** menu option to open the *Search and Rename Signals* dialog.
- Enter **Test.** in the *Old Name* edit box.
- Enter **Test_** in the *Signal Name Replacement* edit box.
- Click **OK** to close the dialog and rename the signals.



Add the Compare Enable Signal:

- Add a new signal called **Compare_Enable** and draw the following waveform that is high between 4ns and 8ns

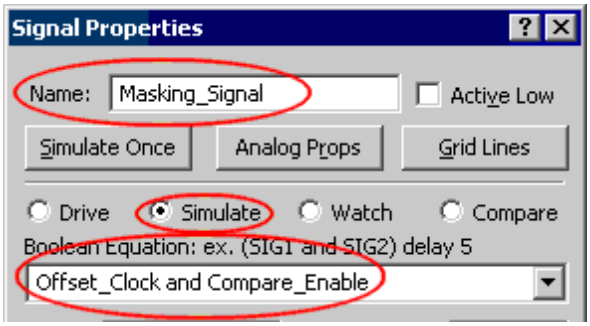


Add the Simulated Masking Signal

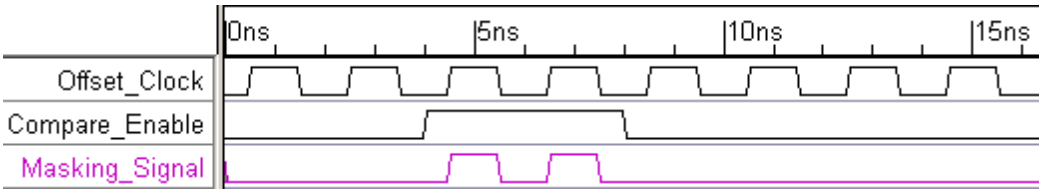
Next, place a simulated signal in the diagram to use as the clocking signal for the comparison:

Copyright © 2014, SynaptiCAD Sales, Inc.

- Add a signal to the diagram and set the following properties in the *Signal Properties* dialog:
- Set the *Name* to **Masking_Signal**.
- Select the **Simulate** radio button (instead of *Drive*).
- Enter **Offset_Clock and Compare_Enable** in the *Boolean Equation* edit box.



- Press the **Simulate Once** button to generate the simulated signal.



- Notice that the masking signal is high only during the times that both the Compare_Enable and the Offset_Clock are high. These are the clock segments that are not masked. The comparison will be performed during these segments.

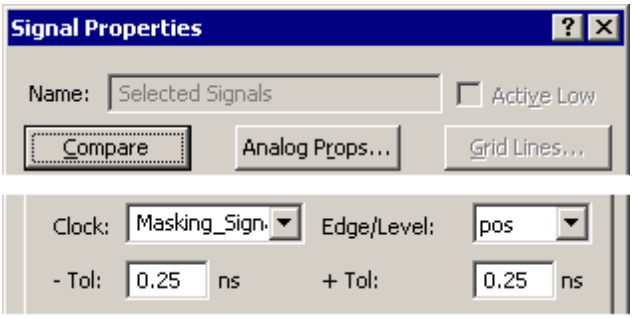
Use the Masking Signal to Mask Clock Segments

Next we will edit the compare signals so that they use the new **Masking_Signal** as a clock.

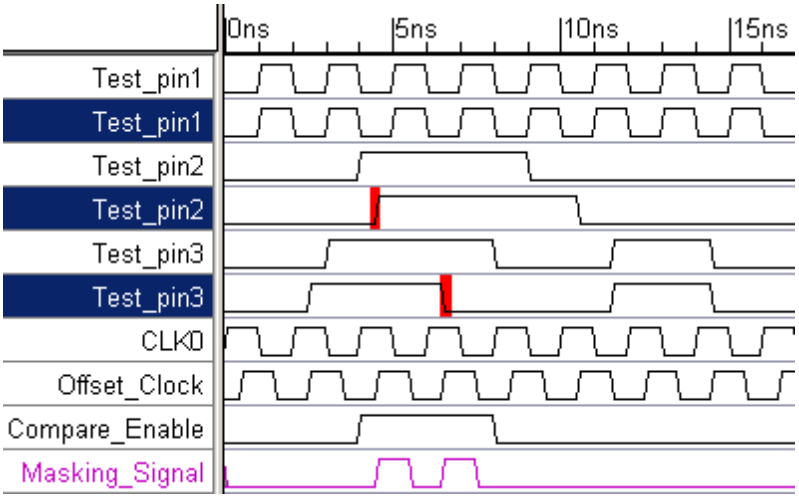
- Either press the **SET ALL** button or choose the **View > Compare and Merge > Edit Compare Signals** menu to select all the compare signals and open the *Signal Properties* dialog.



- Set the *Clock* to **Masking_Signal**.
- Click **Compare** to apply the changes and perform the comparison.



Copyright © 2014, SynaptiCAD Sales, Inc.



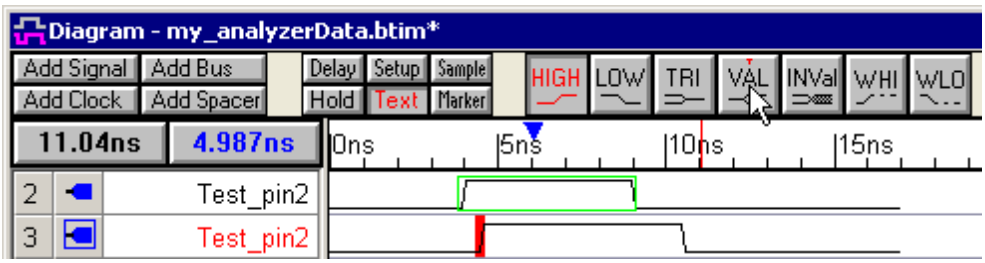
- Notice that the first and last differences that were previously in the diagram have now disappeared, because they are masked from the comparison.

(Compare) 10: Don't Care Regions

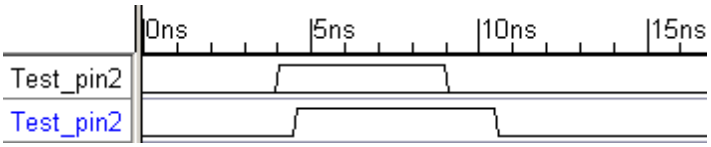
A related function to the masking signals is don't care regions. Sometimes you may not want to compare certain regions in time on a particular signal. One case would be on a bus signal where the actual data did not matter. If you want the compare function to skip a particular section, then just turn the waveform into a valid region on the reference signal.

To create a don't care region:

- On the **test.pin2** reference signal (not the compare signal), click in the high segment to select it.



- Press the **VAL** button in the button bar at the top of the waveform window. This will change the waveform to a valid waveform.
- Press the **Compare** button to apply the changes and perform the comparison.



- Notice that the difference is no longer flagged, because the don't care segment on the reference signal blocks the compare.

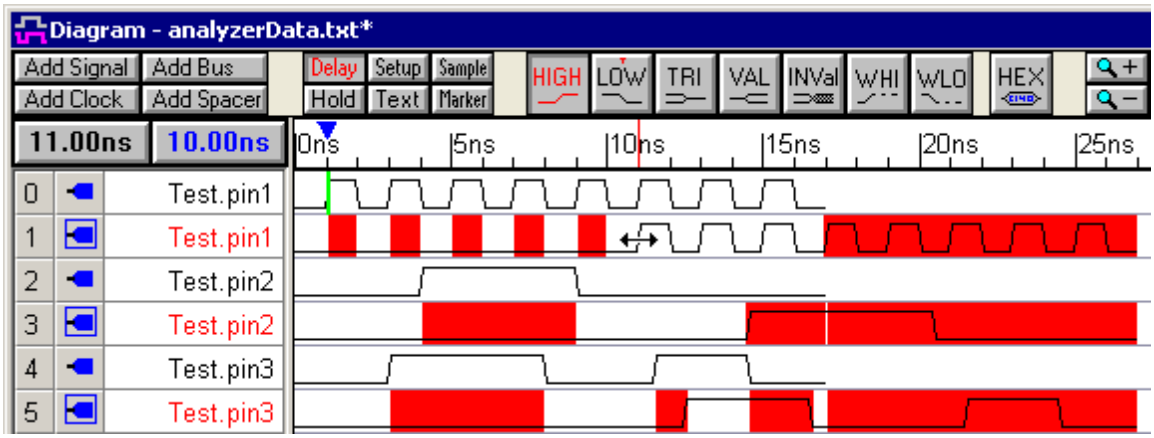
Copyright © 2014, SynaptiCAD Sales, Inc.

(Compare) 11: Adjust the Time Difference Between Two Diagrams

The time difference between two timing diagrams can be easily adjusted using the *Edit Waveform Edges* dialog. In this section we will compare two diagrams that are slightly offset in time.

Compare the two data sets:

- Select the **File > Open Timing Diagram From** menu and load the original **analyzerData.txt** file located in the **SynaptiCAD\Examples\TutorialFiles\WaveFormComparison** directory. Don't forget to set the file type to **Test Vector Spreadsheet/Tektronix (*.txt)**. Note: If you accidentally saved over this file during a previous step, simply delete all the signals except the three Test reference signals, change the names back to dots instead of underlines, and change the valid state on pin2 to a high state.
- Select the **File > Compare Timing Diagram** menu option and select the **simulationResults_offset.btim** to load the file and perform a compare.



- Notice that all three of the compare signals have a longer starting segment than their counterparts. This causes much of the diagram to show as differences, when in reality the two data sets have a time offset.
- Use the mouse and the time readout buttons to determine the amount of time that the diagram is offset from the other (10 ns). Also double click on an edge to open the *Edge Properties* dialog and find the exact times of edges.

To modify the time difference between two diagrams:

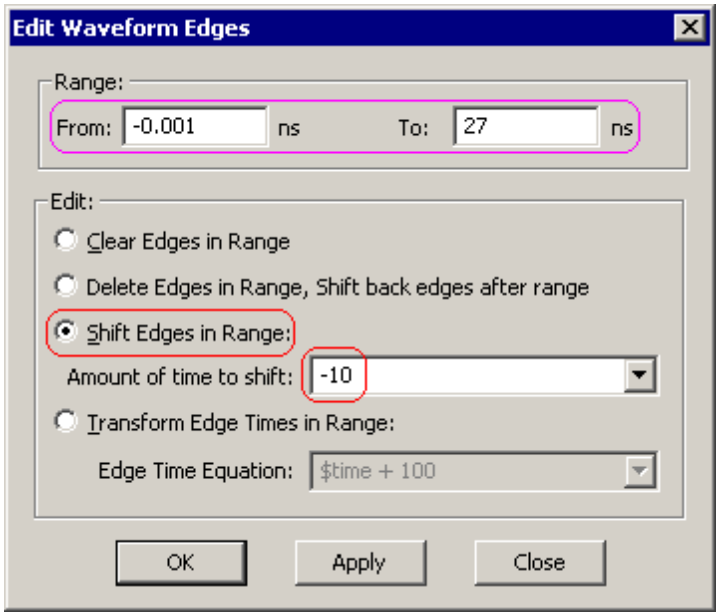
The offset time can either be removed from the compare signals or it can be added to the reference signals. In this tutorial the offset will be removed from the compare signals because the SET ALL button makes it easy to select the compare signals.

- Select all the compare waveforms by pressing the **SET ALL** button. This also opens the *Signal Properties* dialog in group editing mode, but we will not be using it so you can close the dialog.

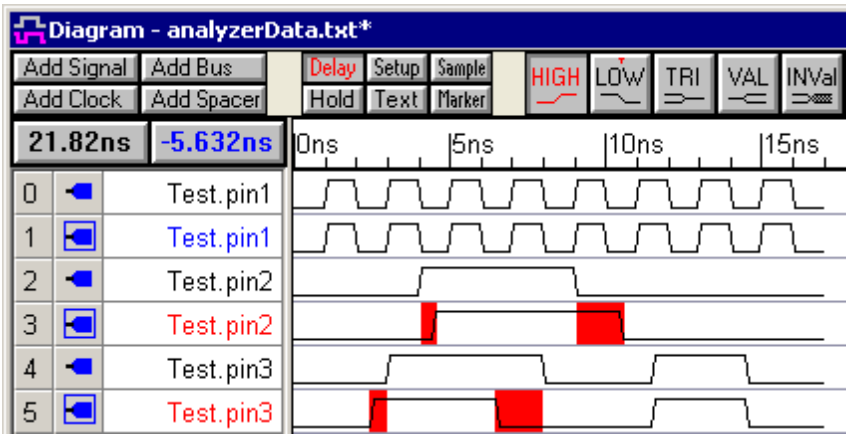


Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **Edit > Edit Waveform Edges** menu option to open the *Edit Waveform Edge* dialog.
- Notice that the time range is the entire timing diagram.
- Select the **Shift Edges in Range** radio button.
- Enter -10 into the **Amount of time to shift** edit box. The time unit, ns, in implied by the display time unit of the diagram.
- Press **OK** to shift the selected waveforms and close the dialog



- Press the compare button to perform another compare.



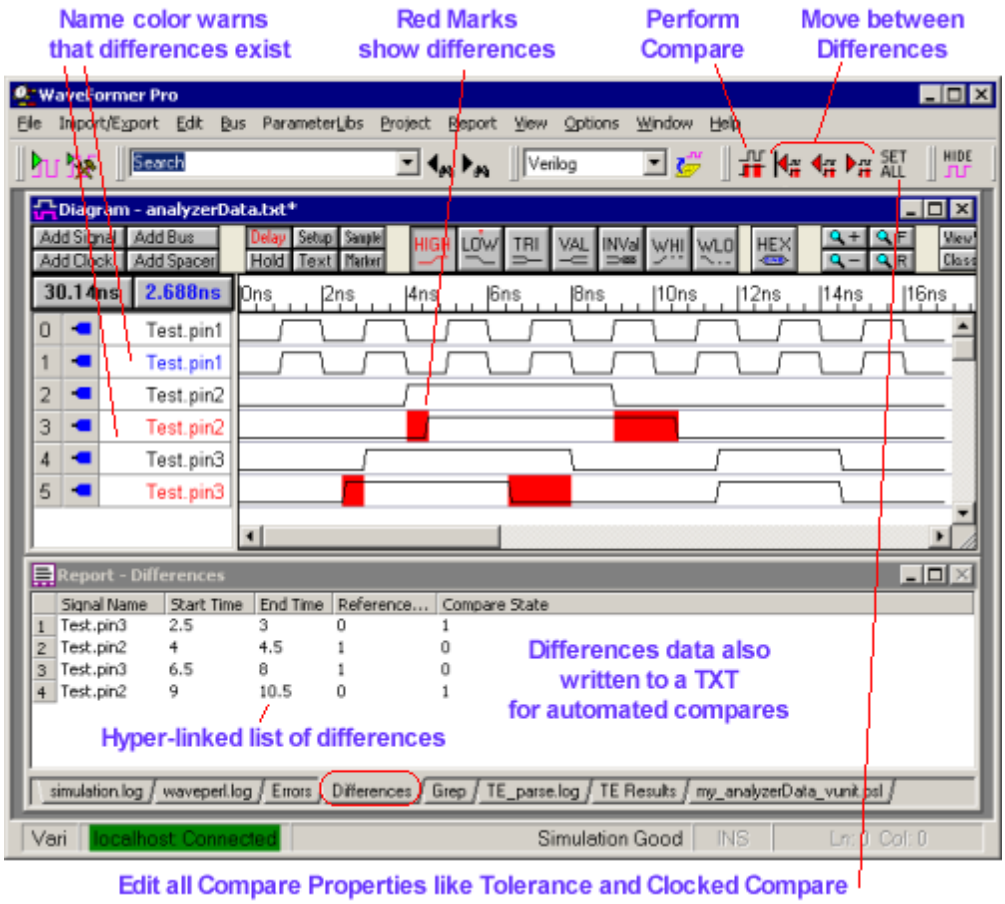
Tip: The *Edit Waveform Edges* dialog can also be used to perform frequency multiplication. See the Timing Diagram Editors manual Section 1.7: Editing Waveform Edges from an Equation for more information on this dialog.

(Compare) 12: Summary of the Comparison Tutorial

Congratulations, you have completed the Waveform Comparison Tutorial! You have compared individual signals by changing the names to match and changing the type of one signal to **Compare**. You have compared two timing diagrams and edited the Tolerance and Clock settings using the **Set All** button. You have also used the Edit Waveform Edges function to adjust the timing of a diagram in order to prepare it for a compare.

As a side note, file comparison can also be done automatically using the batch mode feature discussed in the Timing Diagram Editors manual Section 11.5 Batch Mode.

Copyright © 2014, SynaptiCAD Sales, Inc.

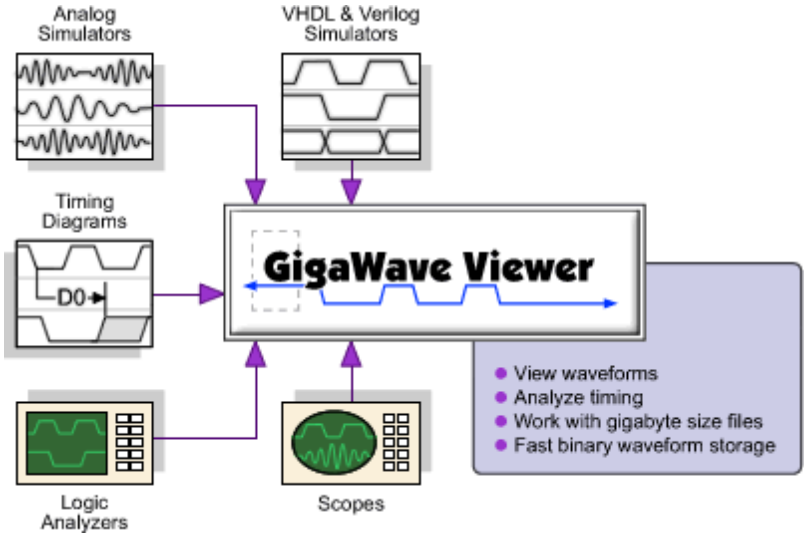


Copyright © 2014, SynaptiCAD Sales, Inc.

Gigawave and WaveViewer Viewer Tutorial

This tutorial covers features that are useful in viewing and analyzing large waveform files. In particular, it covers some of the features in the free waveform viewer, WaveViewer, and the more powerful and faster GigaWave Viewer product. Waveform viewing features are also included in all of the timing diagram editor products and in BugHunter Pro.

This tutorial covers how to convert waveform files into SynaptiCAD's binary waveform format (BTIM format) for faster loading and viewing of the waveform data. For even quicker file loads, a subset of a file's signals can be selectively loaded using either the Import dialog or a filter file. This tutorial also demonstrates how to browse large files using the Show/Hide dialog and by searching for signals and states.

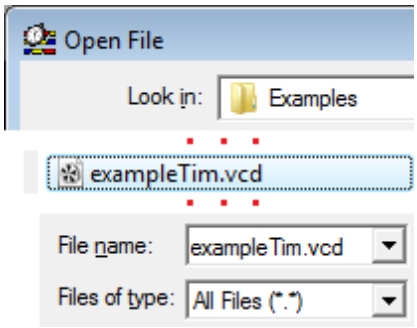


(Viewer) 1: Converting a vcd file into a btim file

When viewing non-native waveform formats, such as VCD files, we recommend first converting the file to SynaptiCAD's native BTIM format. The resulting compressed BTIM file will generally be around 200x smaller than the original file and will load much faster (for example, a BTIM file will typically load around 500x faster than an equivalent VCD file). To conversion a file, just load the file into the viewer, then save it to the BTIM format.

Open the example file "exampleTim.vcd":

- Choose **File > Open Timing Diagram** menu option to launch the *Open File* dialog.
- In the **File of type** drop-down, choose **Verilog Value Change Dump**.



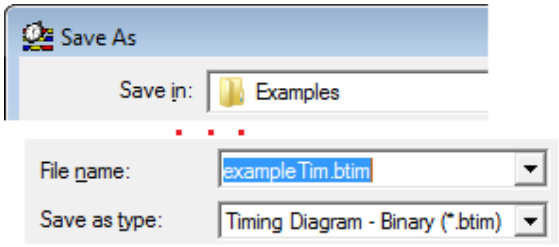
- Select the file **C:\SynaptiCAD\Examples\exampleTim.vcd**.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Click the open button to load the diagram.

Save the waveform file to the BTIM format:

- Choose **File >Save As** menu option to open the *Export Timing Diagram As* dialog.
- In the **Save as type** drop-down, choose **Timing diagram – Binary**. This will change the filename to ***exampleTim.btim***.



- Click **Save** to close the dialog box and create the fast BTIM version of the file.
- Close the diagram so we can load a subset of the signals in the next step.

(Viewer) 2: Importing a subset of the Waveforms

In the previous section, we loaded an entire waveform file using the **File > Open** menu. It is also possible to load a subset of the waveforms in a file by using the **Import/Export > Import Timing Diagram** menu.

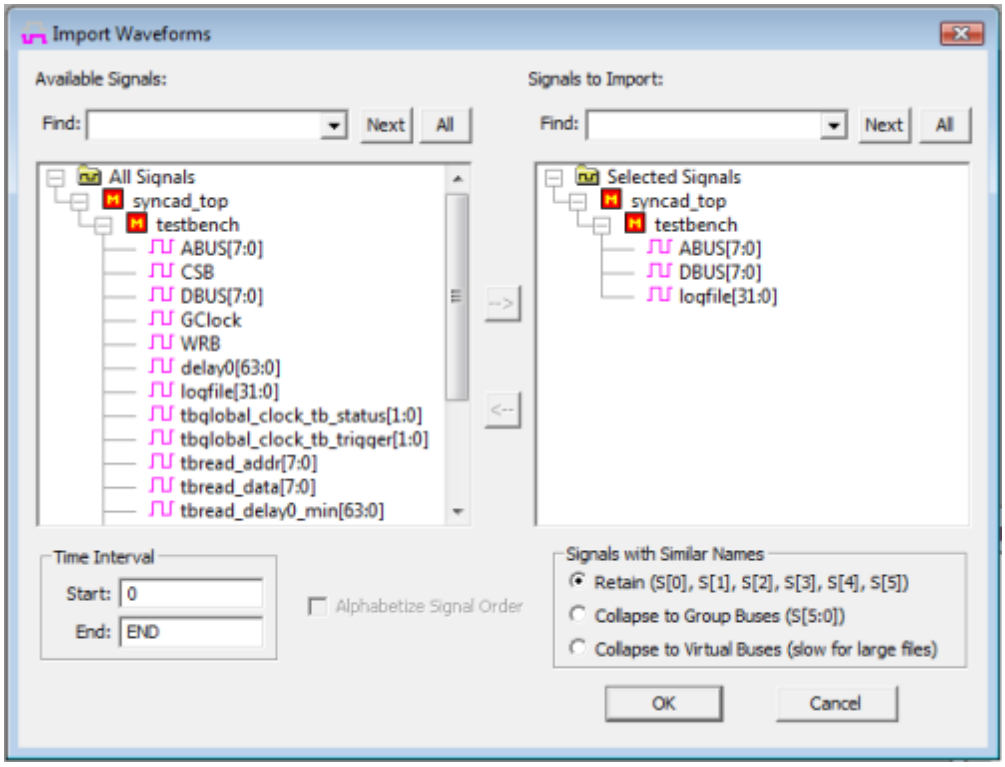
Use Import Timing Diagram to load a subset of waveforms from a file:

- Choose the **Import/Export > Import Timing Diagram From** menu to open the *OpenFile* dialog.
- Select the BTIM file created in the previous section:

  ```
  c:\SynaptiCAD\Examples\exampleTim.btim
  ```

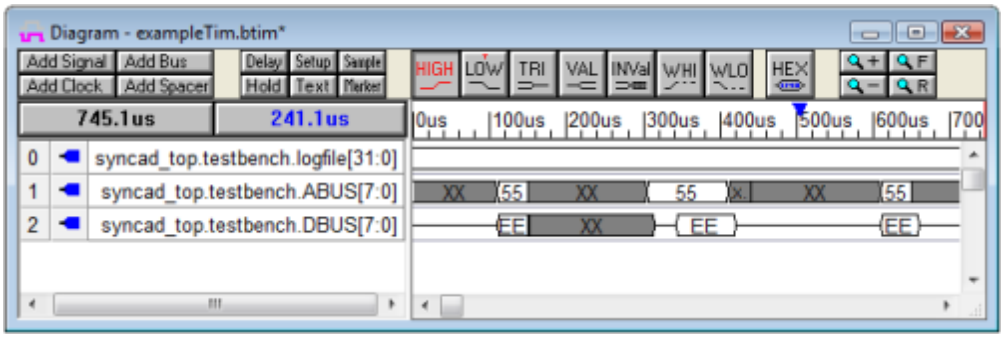
- Press the **Open** button to close this dialog and open the *Import Waveforms* dialog.

Copyright © 2014, SynaptiCAD Sales, Inc.



- In the *Import Waveforms* dialog, move the **ABUS[7:0]**, **DBUS[7:0]**, and **logfile[31:0]** signals from the **Available Signals** list to the **Signals to Import** list by selecting the signal names and clicking the -> button.
    - Note: Signals can be moved in groups by pressing the <CTRL> key while selecting several signals at once.
- Notice that the *Import Waveforms* dialog has several useful options that are not covered further in this tutorial:
    - The **Alphabetize Signal Order** will cause the signals to be alphabetized when they are displayed in the viewer. Otherwise the signals will appear in the timing diagram in the same order that they are found in the file they are imported from.
    - The **Collapse to Group Buses** takes groups of similarly named signals (e.g. bus0, bus1, bus2) and creates additional group bus signals to display the data as buses.
    - The **Collapse to Virtual buses** causes all numbered signals (e.g. bus0, bus1, bus2) to be collapsed into virtual buses (bus[2:0]).
    - The **Time Interval** section allows waveform data imported to be limited to only a specific time range.
- Click the **OK** button to close the dialog and load waveforms into WaveViewer.

Copyright © 2014, SynaptiCAD Sales, Inc.

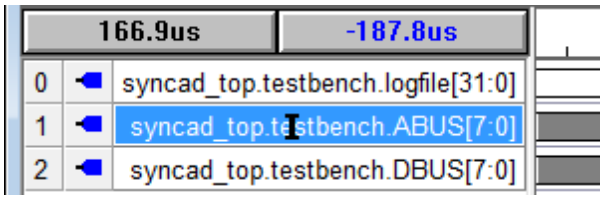


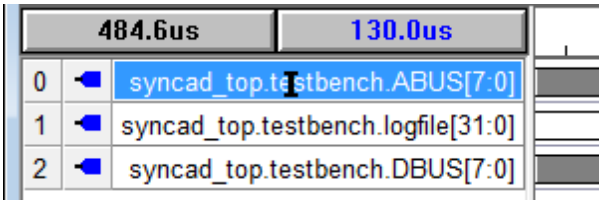
(Viewer) 3: Creating a Filter File to selectively load signals

The **Set Filter File** feature allows you to specify an optional "filter file" that controls what signals get imported from a waveform file (e.g. a VCD file) and the order in which they get imported. This feature is particularly useful when you need to repeatedly view the same subset of signals in several waveform files. The signal properties, such as MSB, LSB, and direction, are also controlled by the signal information that is stored in the filter file. Filter files do not contain any waveform data. Below we will change the ordering of the signals we imported in step 2 and save them into a filter file.

Drag ABUS to the top of the signal list:

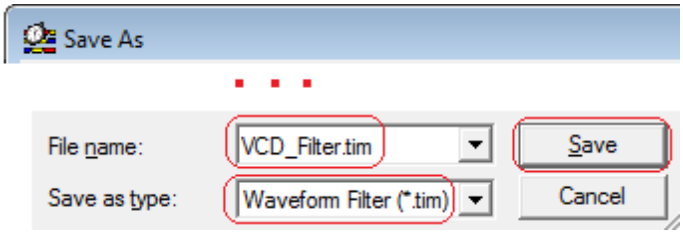
- Drag and drop the ABUS signal label up to the top of the signal list so that the signals are no longer in the default order.





Save the Diagram as a Waveform Filter file:

- Select the **File > Save As** menu option to open the *Save As* dialog.
- In the **Save As type** drop-down, select **Waveform filter (*.tim)** if it is not already selected.
- Type **VCD_Filter** as the name of the filter file in the **File name** edit box.



- Click the **Save** button to create the filter file.

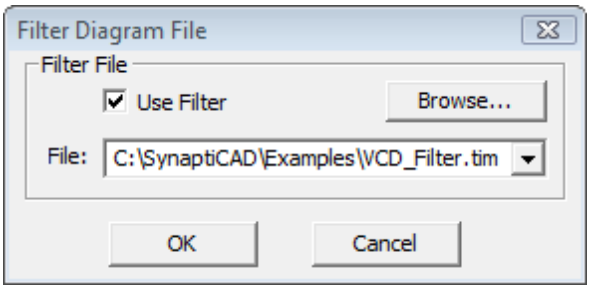
Setup the viewer to use the filter file:

Once a filter file is created, it needs to be set as the current filter file to enable it. Once a filter file is set, each subsequent waveform file load will only load signals that match the filter file signal names.

Copyright © 2014, SynaptiCAD Sales, Inc.

- Select the **Import/Export > Set Filter Diagram file for Import** menu option to open the *Filter Diagram File* dialog.

- Check the **Use Filter** checkbox.
- Select the file by clicking the **Browse** button and finding where you saved it in the previous section.
- Click **OK** to close the dialog.

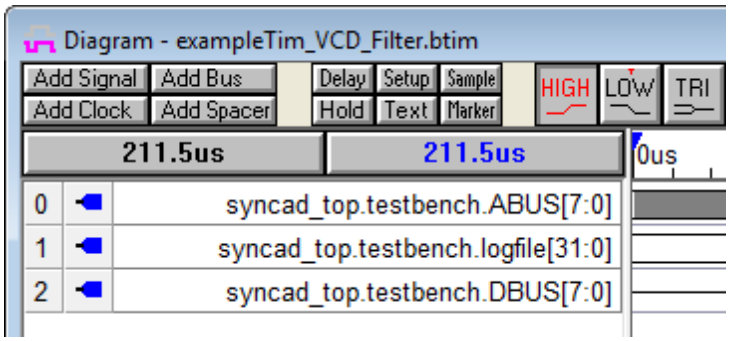


Test the Filter file by loading the original BTIM file:

When a filter file is set, only the filtered signals will be loaded. The filter file will operate on all files that are loaded, regardless of the file format. To try this out, let's load the original BTIM file with the filter enabled:

- Open the **exampleTim.btim** file using the **File > Open timing diagram** menu.

- Notice that only the three buses that were in the filter file have been loaded. And notice that the signals are in the same order as listed in the filter file.



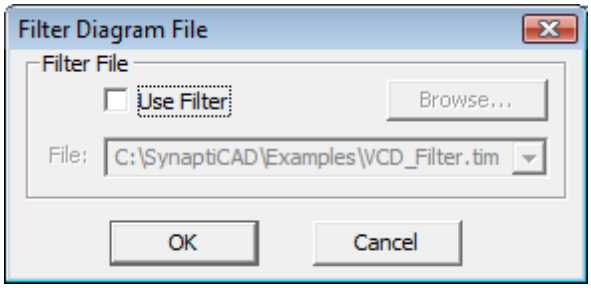
- Also notice that the waveform's filename has been changed to **FileName_FilterName.BTIM**, so that you will not accidentally save this over the original waveform file.

Disabling the Filter file feature:

The filter file affects all files that are loaded, so you may need to turn it off when looking at a new file.

- Select the **Import/Export > Set Filter Diagram file for Import** menu option to open the *Filter Diagram File* dialog.

- Uncheck the **Use Filter** box.



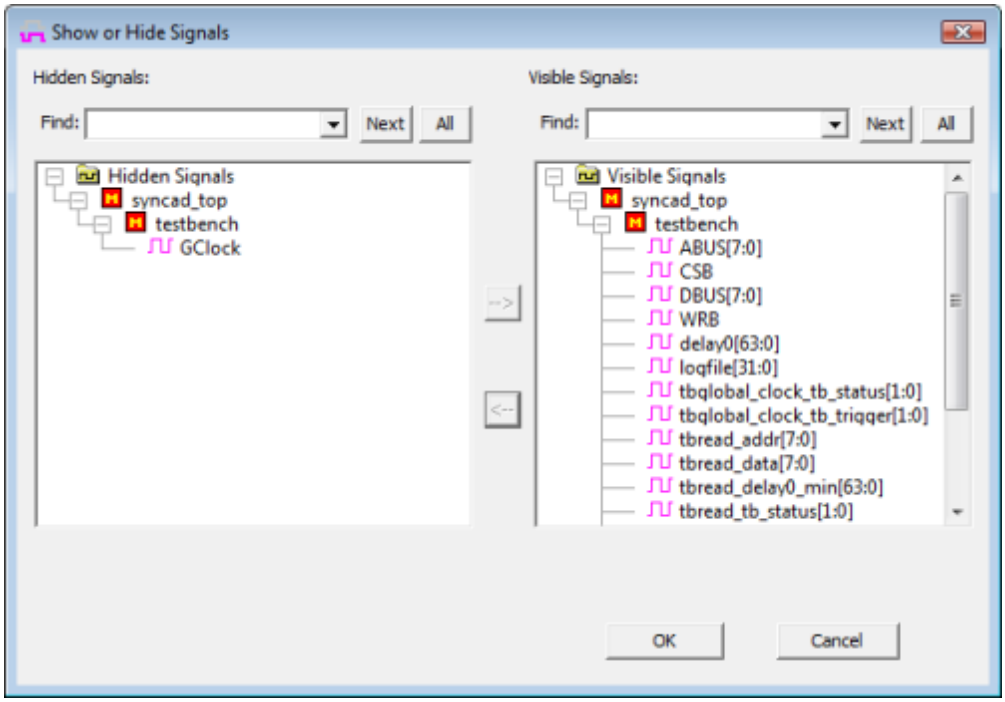
- Close the currently open timing diagram, then choose the **File > Open Timing Diagram** menu and reload the **exampleTim.btim** file again. Note that all signals are now loaded from the diagram because we've disabled the filter file.

Copyright © 2014, SynaptiCAD Sales, Inc.

(Viewer) 4: Show and Hide Signals in the display

Instead of selectively loading signals, it's often easier to load all the signals initially and then hide all but the signals of interest.

- Click on the **View>Show and Hide Signals** menu option to open the *Show or Hide Signals* dialog. The hidden signals are shown on the left side and visible signals (in this case, all signals) on the right.



- Select some signals from the Visible Signals list and press the <-- to move them to Hidden Signals list. Press OK to verify that they are now hidden in the diagram.
- Reopen the dialog and move some of the Hidden signals back to Visible Signals list using the the other arrow button.
- Also practice using the **Find** edit box, to search the signal list to find a particular signal.
- When you are done, unhide all the signals and close the dialog.

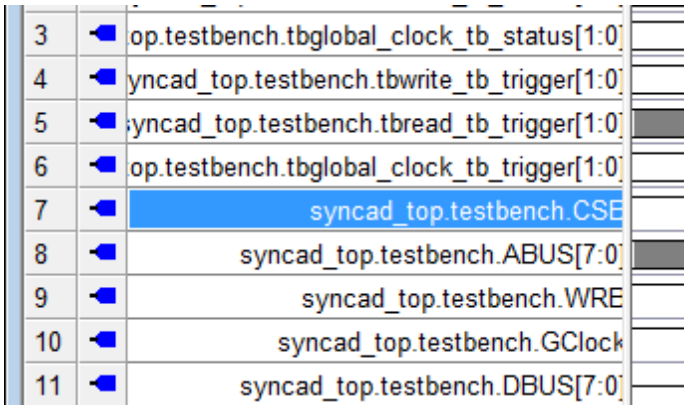
(Viewer) 5: Searching for Signal Names or States

The waveform viewer can search for a particular signal name or a particular signal state using the search box in the main windows tool bar. There are also several mouse techniques for quickly finding times and state values of waveforms and events in the diagram.

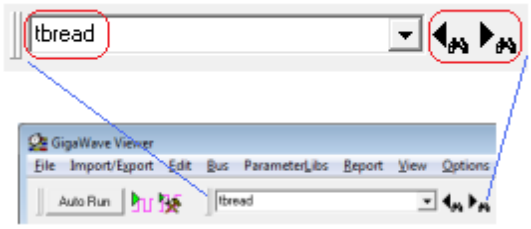
Copyright © 2014, SynaptiCAD Sales, Inc.

Search for a Signal name in waveform window:

- Select a Signal name to indicate the starting point of the search. For this tutorial just pick some signal in the middle of the list.



- Type **tbread** into the Search box on the main window's tool bar.
- Use the Back Search and Forward search to hop to each signal that contains the tbread string in the name.

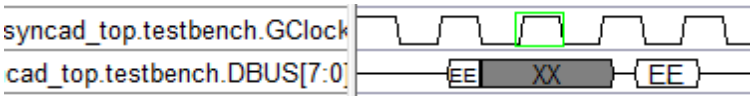


Search for a Signal State in waveform window:

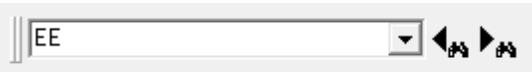
- Press the Zoom Full Button so that you can see all the waveform data for this demonstration.



- Select any waveform edge or state to set the starting point for the search.

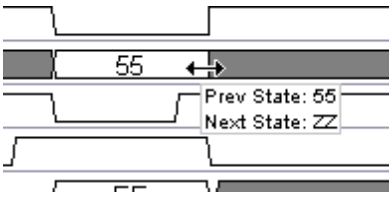


- Type **EE** into the search box and use the next and previous search buttons to find the waveform segments with that state value.



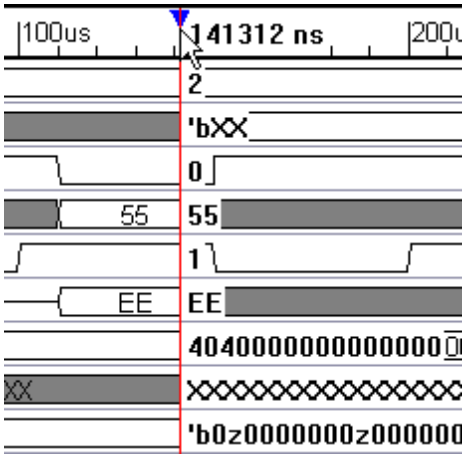
Other techniques for finding waveform state values and times:

- Hover over an edge or waveform segment with the mouse cursor to display the next and previous states.

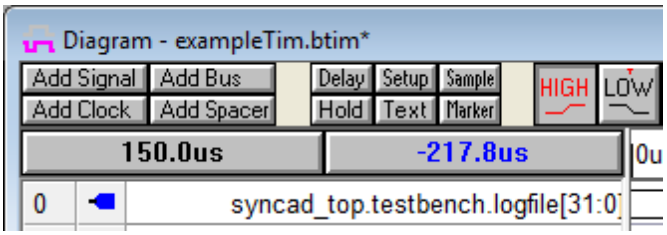


Copyright © 2014, SynaptiCAD Sales, Inc.

- In the Time Line, left click down (and hold) to display a temporary marker with all the states listed.
- A permanent Marker line can be added by adding a Time Marker and checking the **Display Signal States** box. See Chapter 6 in the Timing Diagram Editors manual.



- The Black time button (located above the signal names) indicates the time of the current mouse cursor position.
- The Blue time button shows the time distance between the current mouse cursor position and the last mouse click.



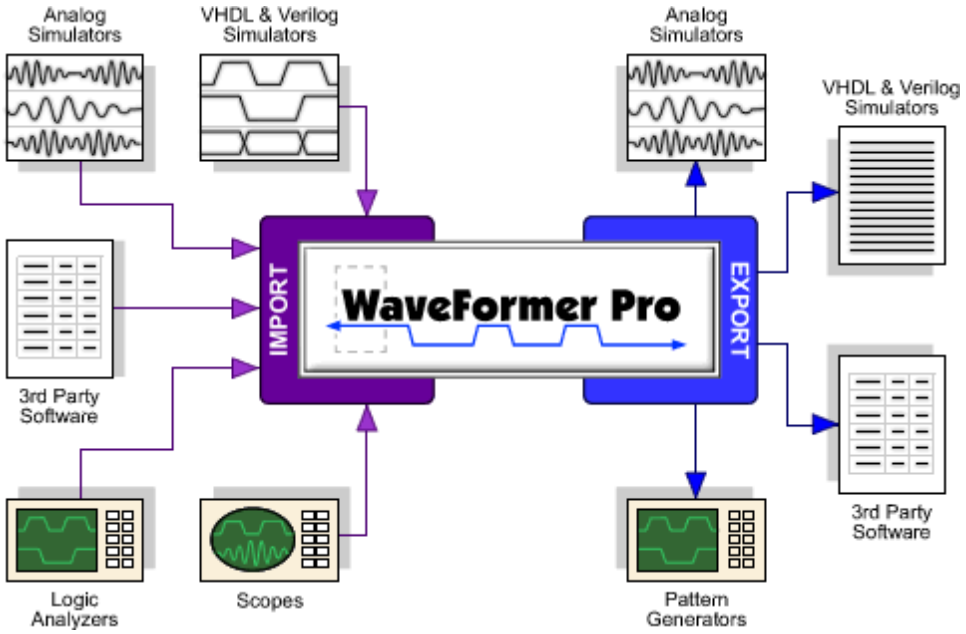
(Viewer) 6: Options: Gigawave, Waveform Comparison,Transaction Tracking

Several optional features can be purchased to extend the power and functionality of the free WaveViewer:

WaveFormer Pro with Pattern Generator Support:

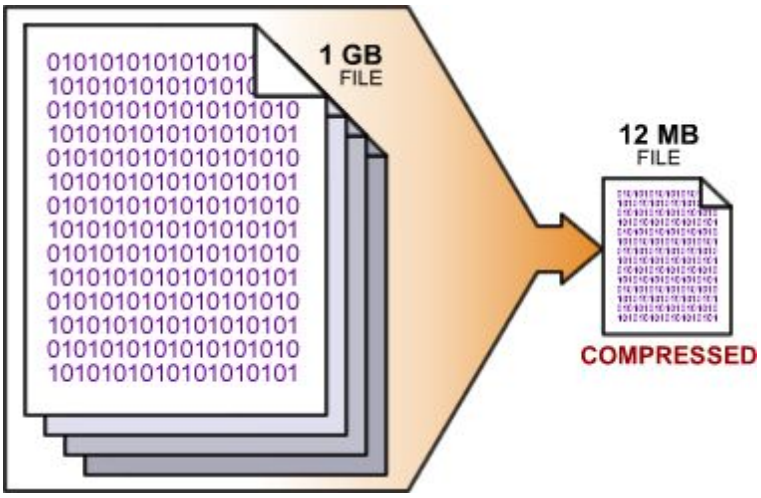
In addition to being a great waveform viewer, WaveFormer Pro supports over 55 different formats for importing and exporting waveform data. Import waveform data from simulators, logic analyzers, and timing analysis programs. Export waveforms to VHDL, Verilog, SPICE, Pattern Generators, ATE, and FPGA tools. WaveFormer Pro is also a full timing diagram editor, so you can create specification diagrams and analyze timing.

Copyright © 2014, SynaptiCAD Sales, Inc.



GigaWave Feature:

The GigaWave feature converts WaveViewer into GigaWave Viewer, a high capacity waveform viewer capable of handling multi-gigabyte VCD files. Without GigaWave, WaveViewer is limited to diagrams of less than 10,000 signals and less than 1 million waveform state changes. The GigaWave feature also comes with a PLI-based library that can be integrated with your favorite simulator to directly generate highly compressed BTIM files (no intermediate dump to VCD is required). Using direct BTIM waveform dumping can speed up simulation by up to 3x compared to the same simulation using an ordinary VCD dump because of the reduction in file I/O, and the resulting files are generally 200x smaller.

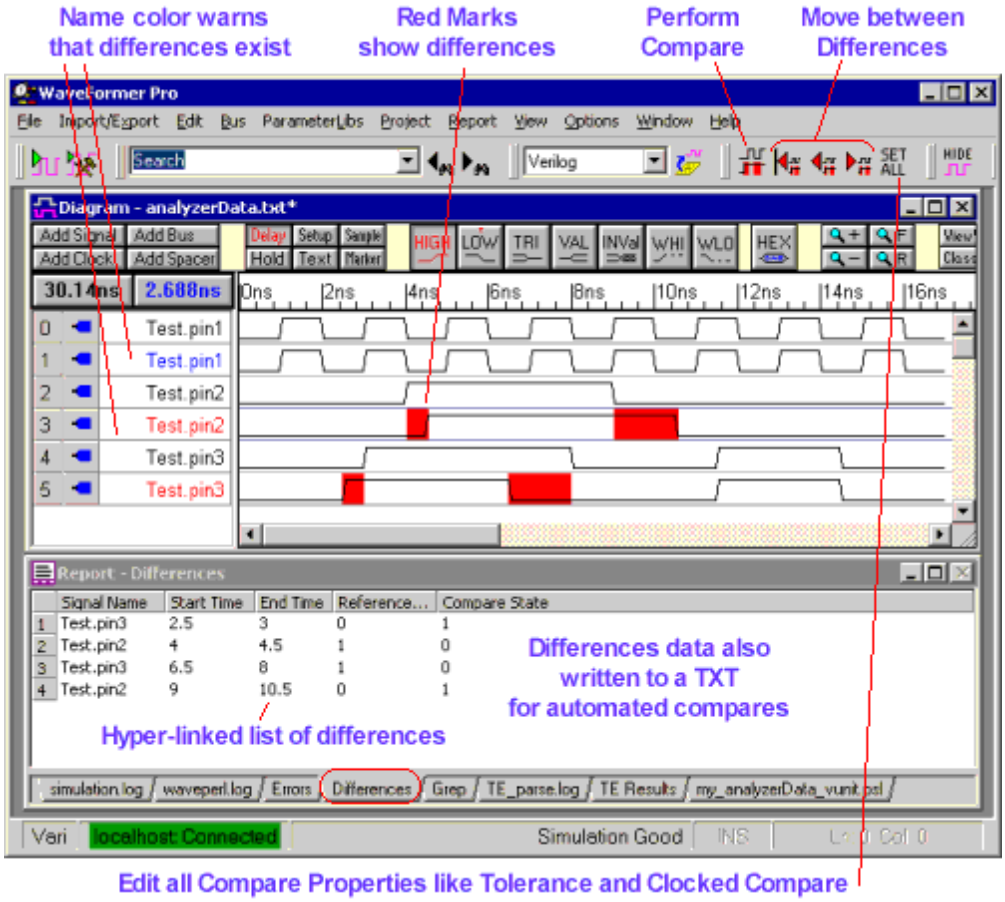


Waveform Comparison Feature:

The Waveform Comparison feature lets users compare waveforms for two timing diagrams or individual signals. This feature is particularly useful for comparing two different simulation runs, as well as for comparing logic analyzer data to a simulation run. The specific regions where waveforms differ will turn red when two waveforms are compared. In addition to standard waveform comparison, where all differences are detected, the comparison feature also supports "Clocked comparison" where

Copyright © 2014, SynaptiCAD Sales, Inc.

waveforms are only checked at clock edges. Tolerances can also be specified to determine what constitutes a significant difference between two waveforms. For more information on using the Waveform comparison feature, please refer to Chapter 9 of the Timing Diagram Editor Manual.



Transaction Tracking Feature:

The Transaction Tracking feature lets users view waveform data as "transactions" instead of as just signals. The user specifies PSL Sugar expressions that describe what transactions to search for, and the transaction tracker search engine finds matching transaction records and displays them graphically in the timing diagram window.

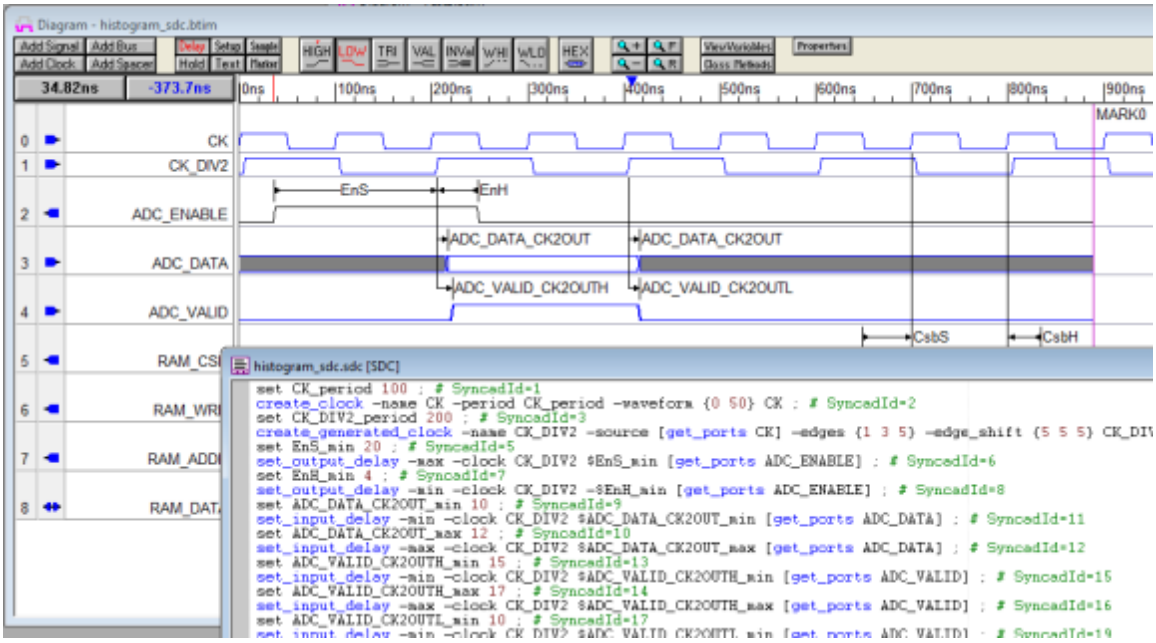
(Viewer) 7: Waveviewer/GigaWave Viewer Tutorial Summary

Congratulations! You have completed the WaveViewer and GigaWave Viewer tutorial. In this tutorial we introduced various techniques for opening, importing, saving and converting .vcd files into btim files for faster file loading. We created a filter file and used it to selectively load signals (filter files can also be used to reorder signals and override the default properties on the loaded signals). We also covered the differences between GigaWave Viewer and WaveViewer and briefly discussed the optional VHDL/ Verilog/SPICE export, Waveform Comparison, and Transaction Tracking features. More information on these topics can be found in the online help.

Copyright © 2014, SynaptiCAD Sales, Inc.

SDC Timing Generation Tutorial

This tutorial covers generating SDC timing constraints from a timing diagram. We will be modeling the timing surrounding an FPGA in a Histogram Circuit which includes external clocks, controlling an ADC, and working with a Synchronous SRAM. You will need a valid license or evaluation license to complete the tutorial.



(SDC) 1: Setup Project and Create Timing Diagram

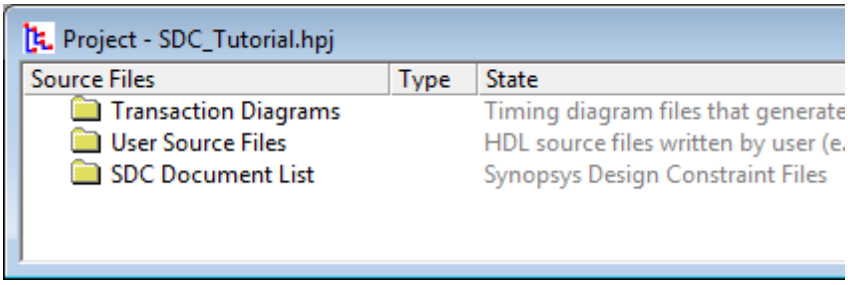
SDC timing files can be generated from any timing diagram, whether it is contained in a Project or just saved as an independent diagram. However, by using a project you will be able to quickly see the generated SDC code and it will make doing this tutorial easier. Not all the SynaptiCAD products have a project window, so there will be alternate instructions for those just using a plain timing diagram.

[Project Users] Creating a Project

WaveFormer with Reactive Testbench option, DataSheet Pro, TestBencher Pro, and BugHunter (or VeriLogger Extreme) all contain the project window. Timing Diagrammer and WaveFormer Pro do not have the project window. If you are evaluating one of the timing diagram products, you can also request a license for the project window. Skip this step and proceed to the "Create a Timing Diagram" step if you do not have a license for a product that supports project creation.

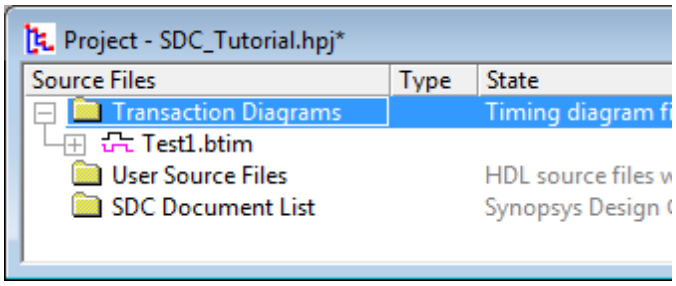
- Choose the **Project > New Project** menu option to open the ***New Project Wizard*** dialog.
- Name the Project, then close the dialog to create the project and open the Project window.

Copyright © 2014, SynaptiCAD Sales, Inc.



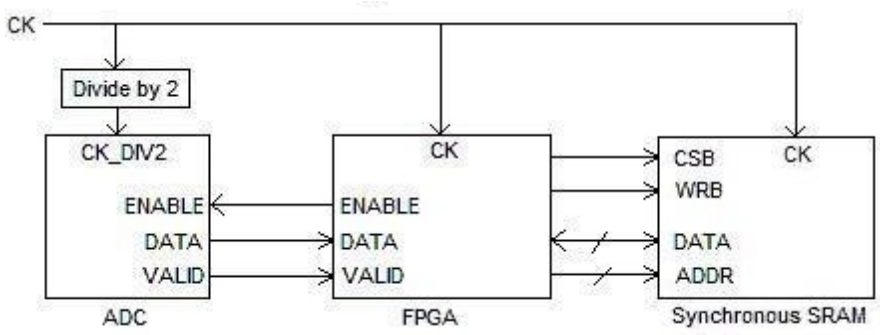
Create a Timing Diagram

- Choose the **File > New Timing Diagram** menu option to create a new timing diagram.
- For Project Users, inside the timing diagram, right click in the signal name pane and choose **Add Diagram to the Project** from the context menu to open a ***Save As*** dialog.
    - Name the diagram and press the **Save** button to add the diagram to the **Transaction Diagrams** folder.



(SDC) 2: Enter Master Clocks

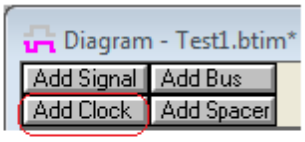
Generally it is a good idea to start an SDC timing diagram by creating the clock signals, since most of the timing will be related to the clocks. In this design example, we have a master clock and a derived clock that is a "divide by 2" of the master clock. Each clock will need to enable SDC code generation.

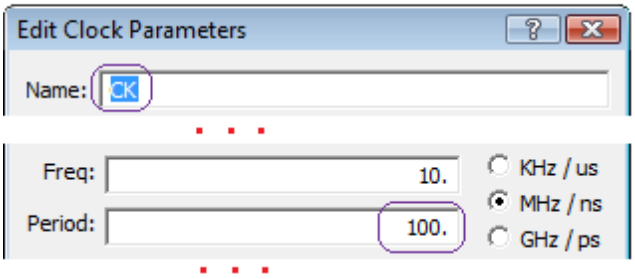


Copyright © 2014, SynaptiCAD Sales, Inc.

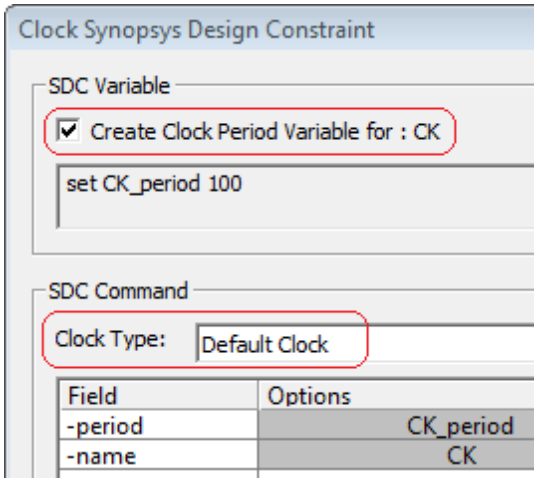
Create the Master Clock:

- Press the **Add Clock** button to create a clock and open the ***Edit Clock Parameters*** dialog.
- Name the clock **CK.**
- Set the period to be **100ns**.
- Clocks do not generate SDC code until you enable the SDC generation. To do this, press the **SDC Code** button at the bottom of the dialog to open the ***Clock Synopsys Design Constraint*** dialog.





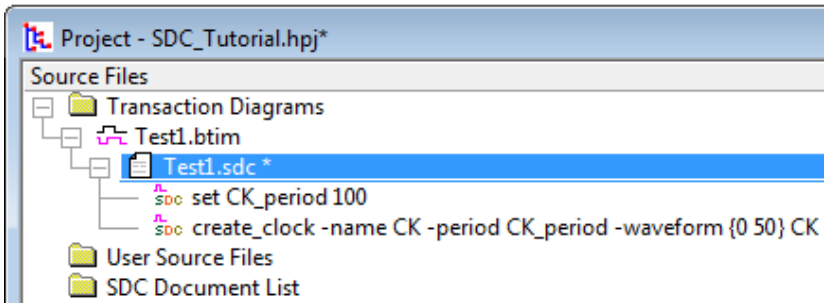
- Check the **Create Clock Period Variable** check box to automatically create an SDC variable called CK_period.
- From the **Clock Type** drop-down choose **Default Clock**. This controls the type of SDC clock command that is generated. A default clock is a clock that attaches to an actual input pin of your design.
- Close the two dialogs.



Looking at the Code In the Report Window:

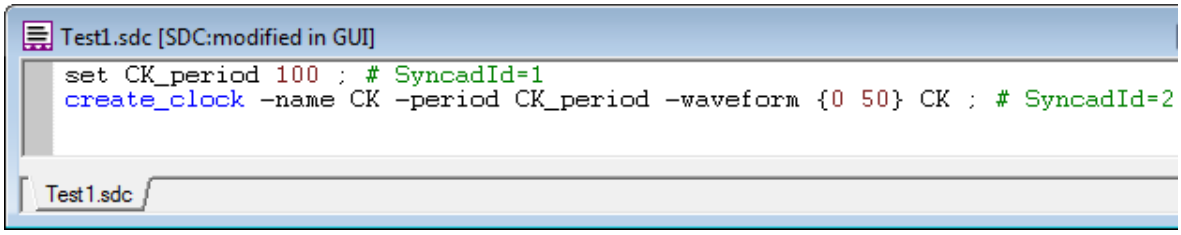
The Code can be viewed and edited in the Report Window. For Project Users just double click on the line in the Project window. For non-project users, open the file separately in the report window.

- In the Project window, notice that you can see the SDC code generated under the timing diagram node.
- Double click on the code to open the code in a Report window.



- For non-project users, save the timing diagram, then choose **Report > Open Report Tab** and pick the file with the **SDC extension** that has the same name as your timing diagram.

Copyright © 2014, SynaptiCAD Sales, Inc.

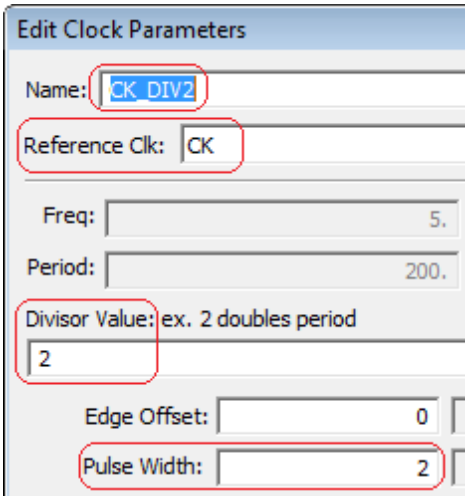


- The SDC code can be directly edited in the Report window and the timing diagram will change to match the code.

(SDC) 3: Enter Derived Clocks

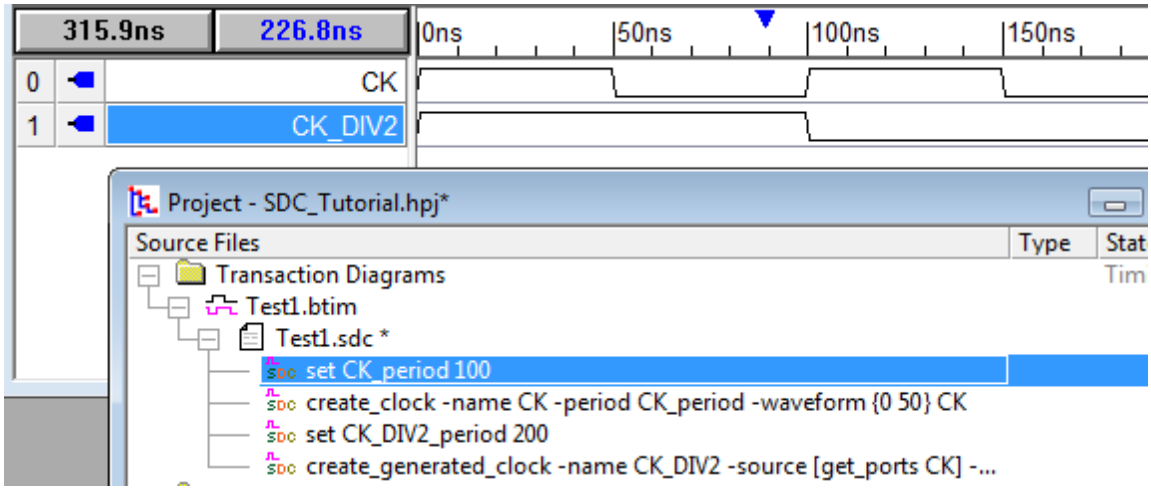
To properly represent the relationship between the master clock and the "divide by 2 clock", we will create a derived clock with the CK clock acting as a reference clock and specify a divisor of 2. Derived clocks produce **created_generated_clock** SDC commands, unlike regular clocks which produce **create_clock** SDC commands.

- Press the **Add Clock** button and create a clock called **CK_DIV2**.
- Choose **CK** as the **Reference Clk** to link the two clocks together.
- In the **Divisor Value** box, type **2** to make this clock a divide by 2 of the master clock.
- Also, change the **Pulse Width** to **2** to make the the derived clock to be high for half the time of the master clock (gives clock a 50/50 duty cycle).
- Press the **SDC Code** button at the bottom of the dialog to open the ***Clock Synopsys Design Constraint*** dialog.



- As before, check the **Create Clock Period Variable** check box to create an SDC variable called CK_DIV2_period, and choose **Default Clock** from the **Clock Type** drop-down box to generate an SDC **create_generated_clock** command that uses this variable. Close the two dialogs.
- You can see the newly generated code in both the Report and the Project windows.

Copyright © 2014, SynaptiCAD Sales, Inc.



(SDC) 4: Sketching the Waveforms

At this point in the tutorial you can either continue to add the signals and parameters as described or you can open a partially completed timing diagram and make changes to enable SDC code generation.

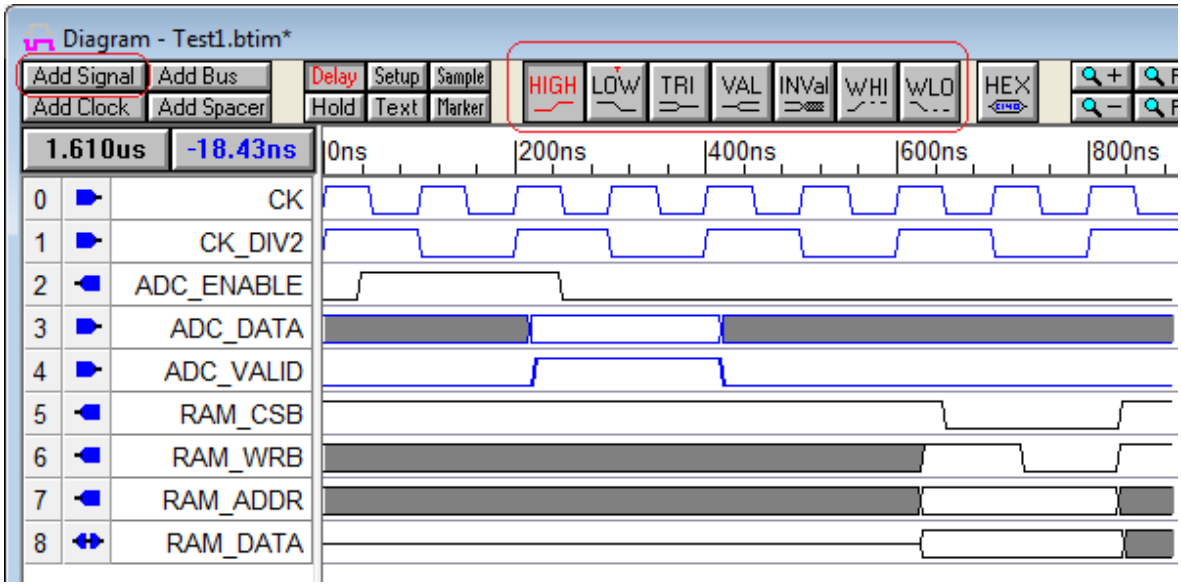
TIME SAVER: Load Histogram_sdc.btim:

- Choose **File > Open Timing Diagram** and open **histogram_sdc.btim** located in the **SynaptiCAD > Examples > TutorialFiles > SdcTutorial** directory.
- For Project Users: In the timing diagram, right click in the signal name pane and choose **Add Diagram to the Project** from the context menu to add the diagram to the project.
- For Non-Project Users: Choose **Report > Open Report Tab** and pick the file with the **SDC file** extension that has the same name as your timing diagram (e.g. histogram_sdc.sdc or test1.sdc).

Sketch the Waveforms:

Next Sketch the waveforms for the ADC and Synchronous SRAM control signals.

Copyright © 2014, SynaptiCAD Sales, Inc.



- Add the above signals and sketch the waveforms. The first Timing Diagram Editor: Basic Drawing Tutorial 10 gives detailed instructions on the drawing environment. For review:
  - The **Add Signal** button adds signals.
  - Double clicking on a signal name opens a dialog where you can change the name and set the direction.
  - Left clicking to the right of the last waveform will draw a waveform section of the type that is selected on the waveform buttons. The waveform buttons toggle between the red one and the one with the little red T above the name.
  - Move signal edges by dragging and dropping them.
  - Change a waveform segment by selecting it then left clicking on a new waveform type.

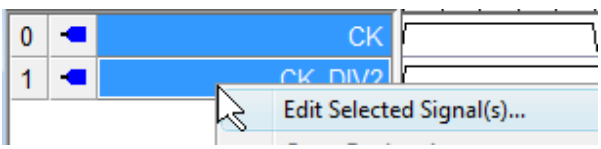
(SDC) 5: Signal Direction

Signal Direction does not directly affect SDC code generation. However, it does affect testbench generation and it adds a bit of readability to the diagram because the little blue icons before the signal name indicate direction. This can be helpful as a visual check in the next steps, because when creating SDC constraints you will generally be adding delays to inputs of your design and setup/holds to the outputs of your design.

By default, all signals and clocks have a direction of output (an output from the timing diagram, in this case from the FPGA's perspective). So if you are drawing the timing diagram, you would might want to change some of the signals to inputs and bidirectional signals by changing the **Direction** box in the ***Signals Properties*** dialog. There is a quick way to select multiple signals and change the directions at the same time, which we will now demonstrate with the clocks.

Example of multiple editing:

- Select both clock signals by pressing the <**Ctrl**> key while left clicking on the clock names.
- Then **RIGHT** click and choose **Edit Selected Signals** from the context



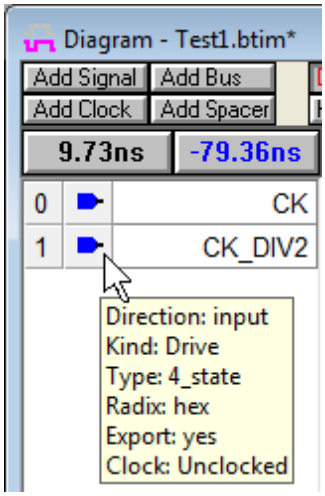
Copyright © 2014, SynaptiCAD Sales, Inc.

menu.

- Notice that the **Name** box in the ***Signal Properties*** dialog is greyed and says *Selected Signals*. This indicates that multiple signals are being edited.
- Choose **Input** from the **Direction** drop-down near the bottom of the dialog.
- Close the dialog.



- If you hover over the icon to the left of the clock name you can see that the signal is now set to Input. Also the little blue icons are pointing in a different direction.



(SDC) 6: Add Delays to Inputs

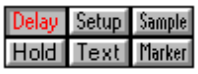
In order for input signals to be properly clocked into the design, we must tell the synthesis tool how much delay has been added to input signals by external logic relative to our clocking signal. In other words, the timing analyzer for the synthesis tool can only extract part of the timing path for any timing path that begins on an input to the design, so the missing part must be given to it by a **set_input_delay** constraint.

In the histogram circuit, the ADC's Data and Valid signals are inputs to the FPGA. From the ADC's data sheet you would find the delay values relative to the clock signal.

For single-bit signals, like ADC_VALID, there will usually be different values for Low-to-High and High-to-Low transitions so it is important to look for that. For buses such as ADC_DATA, there is usually only one delay specified that represents the longest of the L-H or H-L transition times. In this section we will add 4 delays from the derived clock to the transitions of Data and Valid. Then we will enable SDC code generation for these delays.

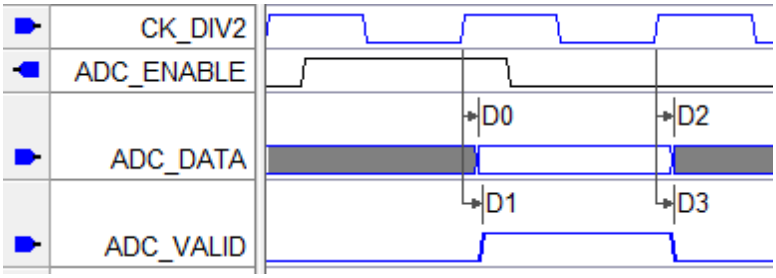
Specify External Delays to Inputs of your Design:

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



Copyright © 2014, SynaptiCAD Sales, Inc.

- Left click on the CK_DIV2 rising edge at 200ns to select it. This edge will be the first or driving edge of the delay.
- Right-click on the **ADC_Data** transition near 200ns to add the delay between the two edges. This is the edge that will be forced by the delay.

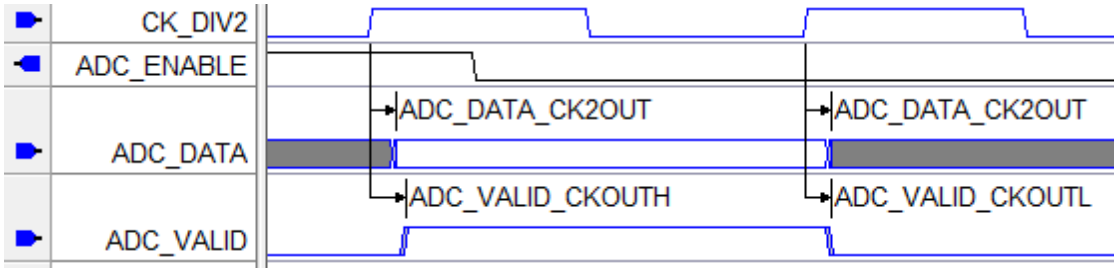


- Add three more delays as shown.
- Double click on each delay and enter the names and min/max times for each delay.

D0 & D2 = ADC_DATA_CK2OUT (10,12)

D1 = ADC_VALID_CKOUTH (15,17)

D3 = ADC_VALID_CKOUTL (10,12)



- Notice that the delay names are also added to the *Parameter* window.

Parameter - Test1.btim*

Add Free Parameter | ADC_DATA_CK2OUT

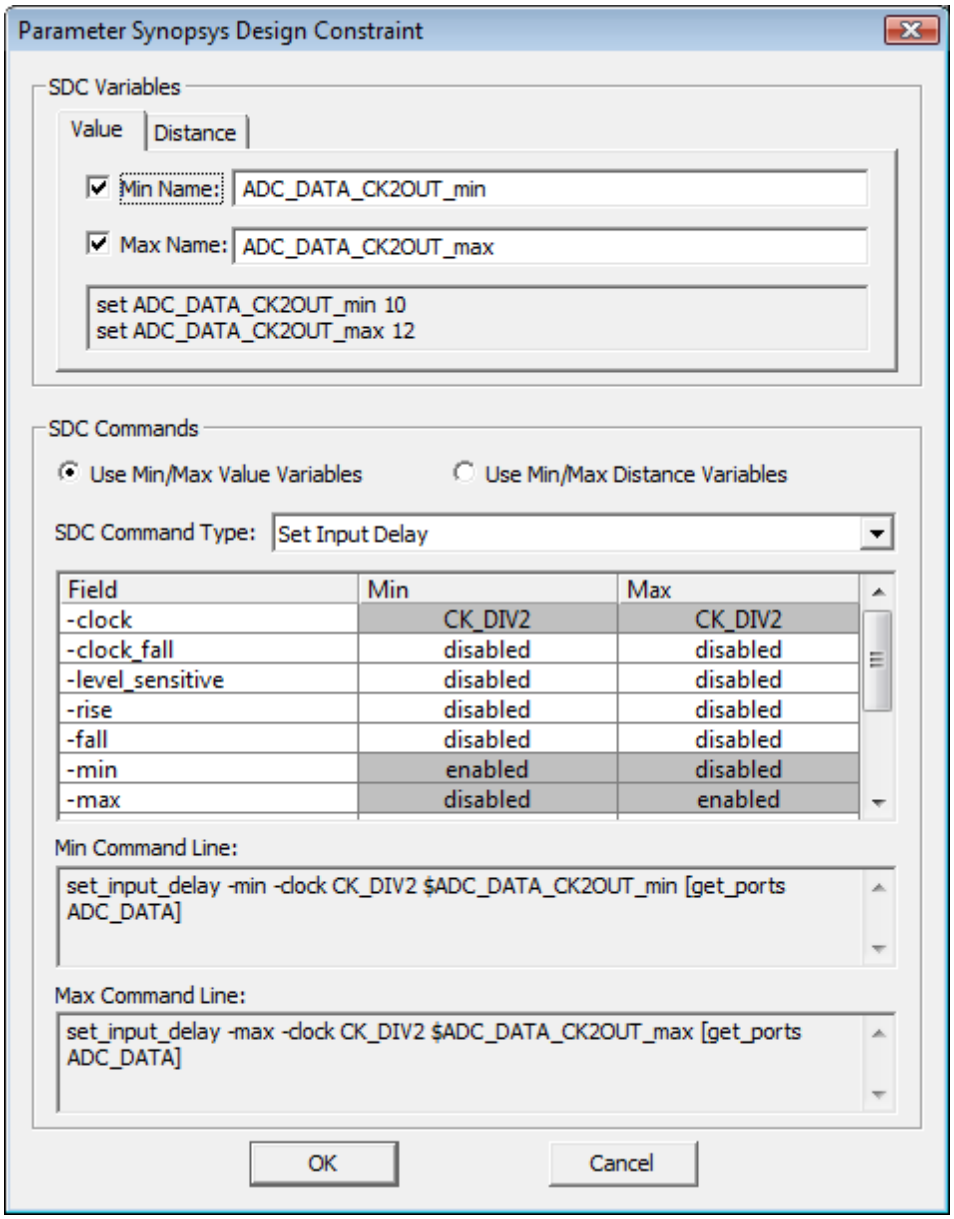
name	min	max	margin
ADC_DATA_CK2OUT	10	12	na (delay)
ADC_VALID_CKOUTH	15	17	na (delay)
ADC_VALID_CKOUTL	10	12	na (delay)
EnS	20		na (free)

Enable SDC Code Generation:

- Double click on a delay to open the ***Delay Properties*** dialog and press the **SDC Code** button to open the *Parameter Synopsys Design Constraint* dialog.
- Create two SDC variables by checking **Min Name** and **Max Name** checkboxes. The two new variables will display in the display area below the checkboxes.
- Create two SDC commands by changing the **SDC Command Type** from **None** to **Set Input Delay**. The two new SDC commands will display in the area below the SDC command parameters grid. Note that the generated SDC commands reference the Min/Max SDC variable names created in the previous step.
- The command parameters grid can be used to set optional parameters to the SDC commands, but this is generally not necessary. Gray fields in the options grid indicate parameters that are

Copyright © 2014, SynaptiCAD Sales, Inc.

computed from the delay timing parameter's data, so only the white fields can be edited.



(SDC) 7: Add Setups and Holds to Outputs

Next we will generate SDC commands to ensure that the design meets the setup and hold time requirements on it's output ports. First we will graphically add the setup/hold requirement for each output port, then we will enable SDC command generation for the setups and holds.

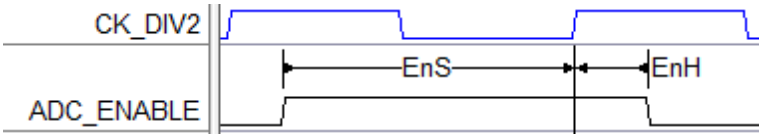
Add Setups and Holds:

- Press the **Setup** button so that right clicks will add setups.



Copyright © 2014, SynaptiCAD Sales, Inc.

- Left click on one of the rising edges of the **ADC_ENABLE** signal to select it.
- Right click on the second rising edge of the **CK_DIV2** to add a blank setup between the selected edge and this one.



- Notice that the arrows of the setup are pointing to the control signal. This means that you added the setup correctly.
- Add the rest of the setups and holds as shown. Always start on the control signal and end on the clock signal when adding setups and holds.

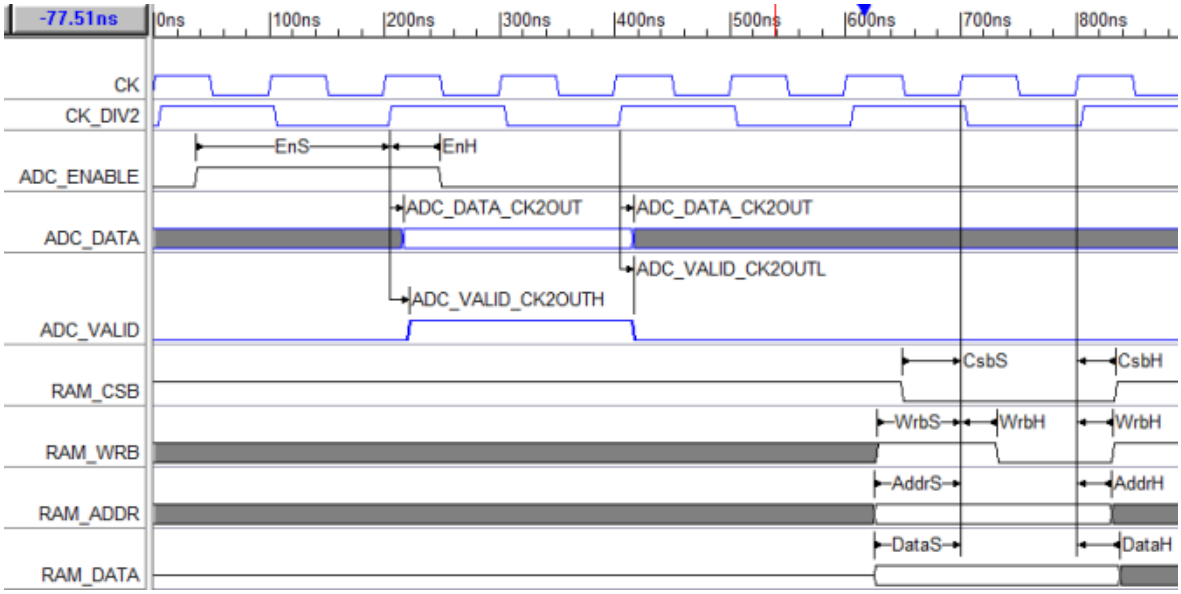
EnS = 20 EnH=4

CsbS=3 CsbH=1

WrbS = 5 WrbH=0

AddrS=5 AddrH=1

DataS=5 DataH=0

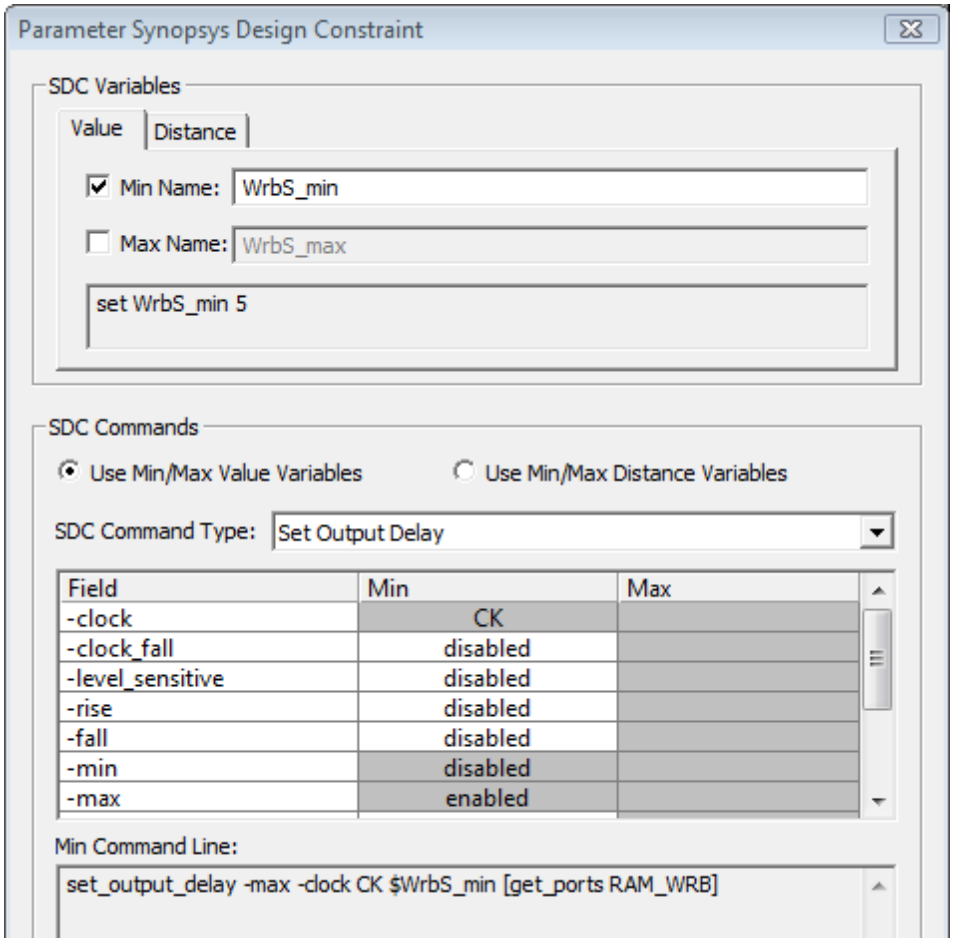


Enable SDC Code Generation

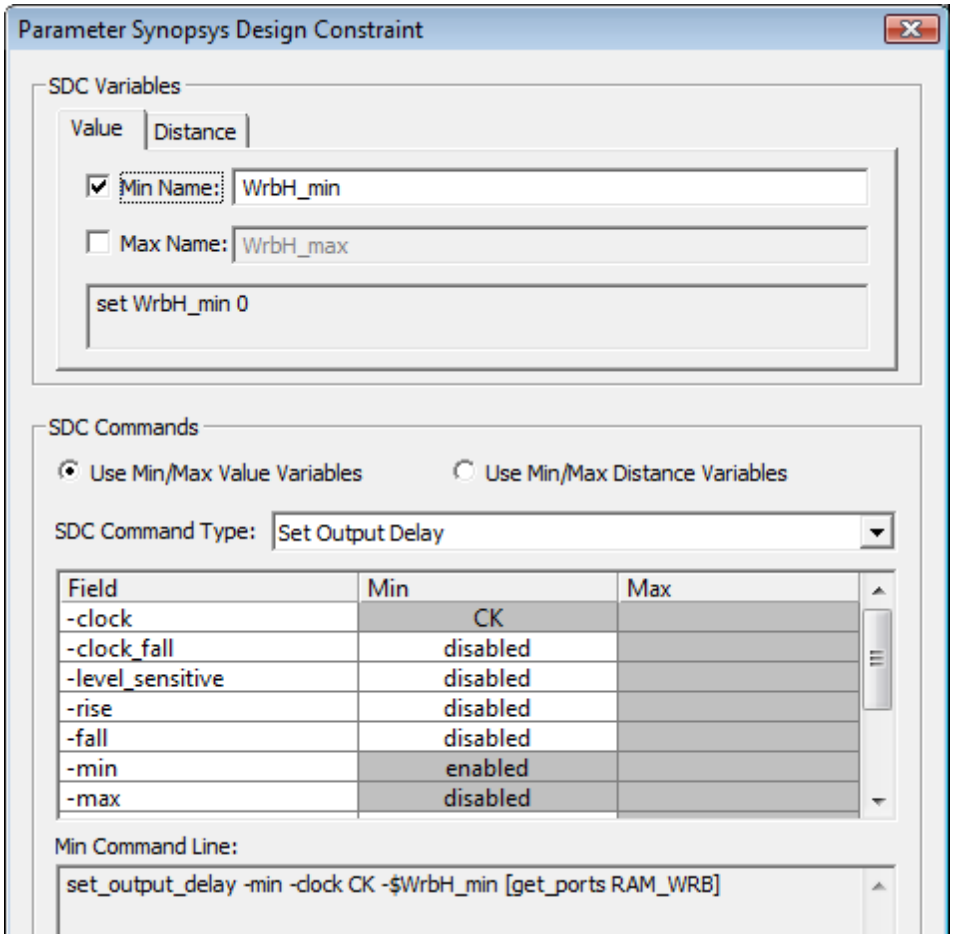
- Double click on a setup to open the *Setup Properties* dialog and press the **SDC Code** button to open the *Parameter Synopsys Design Constraint* dialog.
- Create an SDC variables by clicking the check box for **Min Name**. A new variable will appear in the display area below the checkboxes. Typically setups and holds only have a min value, so it's very rare that you will ever check the **Max Name** for a setup/hold parameter.
- Create an SDC command from the Setup's data by changing the **SDC Command Type** from **None** to **Set Output Delay**. The generated SDC **set_output_delay** command will display in the area below the SDC command parameters grid. Note that the generated SDC command references the Min/Max SDC variable name created in the previous step.
- Enable the Min Name and the SDC Command for the remaining setup and hold times in the diagram.

Copyright © 2014, SynaptiCAD Sales, Inc.

- The results should look similar to the images below. Note that setups generate a "-max" output delay and holds generate a "-min" output delay (because of SDC syntax, "-min" times are specified as negative values, hence the -$WrbH_min in the SDC command for the hold).



Copyright © 2014, SynaptiCAD Sales, Inc.



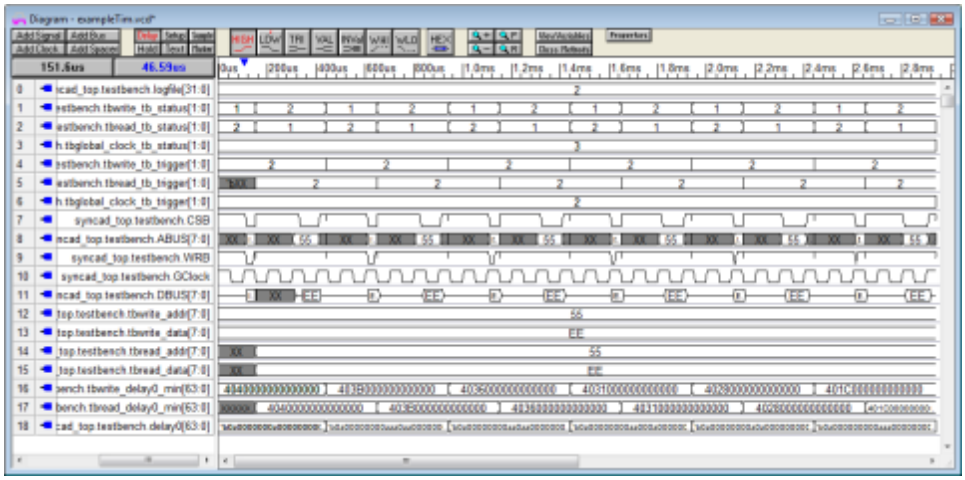
(SDC) 8: VCD Simulation Files Generate SDC

SDC constraints can be derived directly from the timing information contained in simulation or logic analyzer waveforms. The process is similar to the previous steps, loading a waveform file and adding delays, setups, and holds. The difference is that since the timing of the simulated or captured waveforms is exactly the timing for the circuit, you can skip the step of entering values for the parameters by choosing to use "distance variables" rather than "value variables".

Load in a VCD waveform file generated by a Simulator:

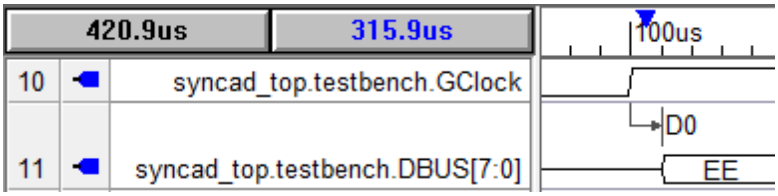
- Choose the **Import/Export > Import Timing Diagram From** menu to open the *Open File* dialog.
- Open the **exampleTim.vcd** file located in the **C:\SynaptiCAD\Examples** directory.

Copyright © 2014, SynaptiCAD Sales, Inc.

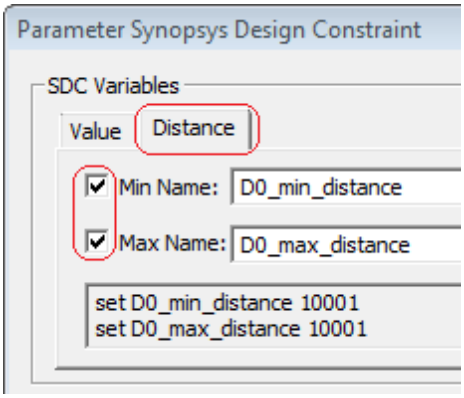


Add a Delay and enable the SDC Distance Variables:

- Zoom into the diagram, and add a delay between the **GClock** rising edge and the start of the **DBUS** cycle.

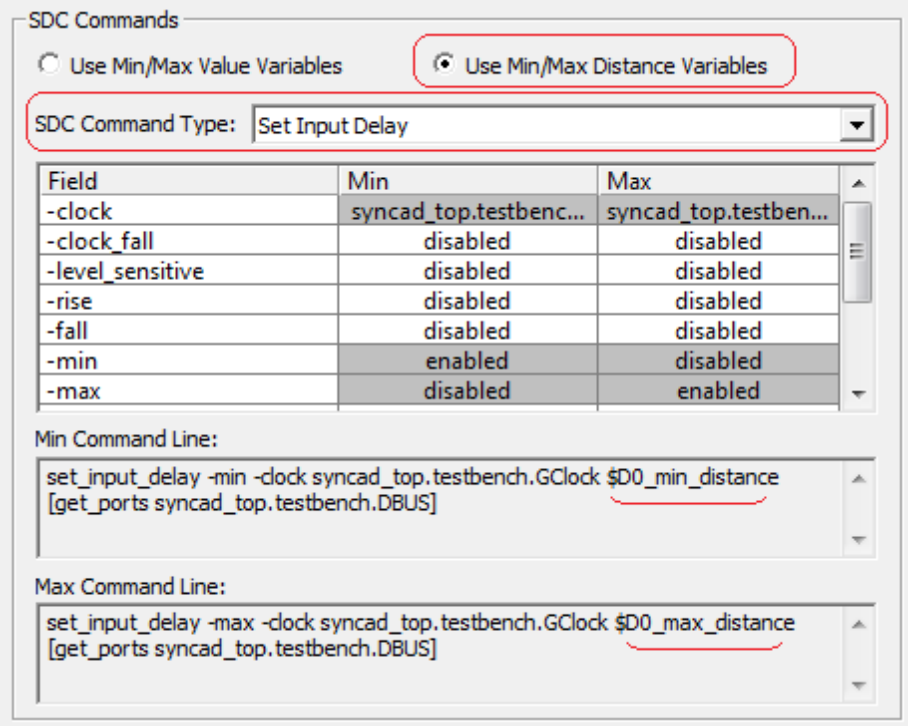


- Double click on the Delay to open the *Delay Properties* dialog.
- Press the **SDC Code** button to open the *Parameter Synopsys Design Constraint* dialog.
- Press the **Distance** tab.
- Check the **Min Name** and **Max Name** check boxes to create the distance variables. These will take the actual value of the distance between the edges (i.e. timing values set by simulation or captured by logic analyzer).



- In the SDC Commands section, check the **Use Min/Max Distance Variables** to use the variables you just created.
- Also you will need to choose the **SDC Command Type** to generate.

Copyright © 2014, SynaptiCAD Sales, Inc.



- Notice that the code at the bottom of the dialog is using the distance variables.

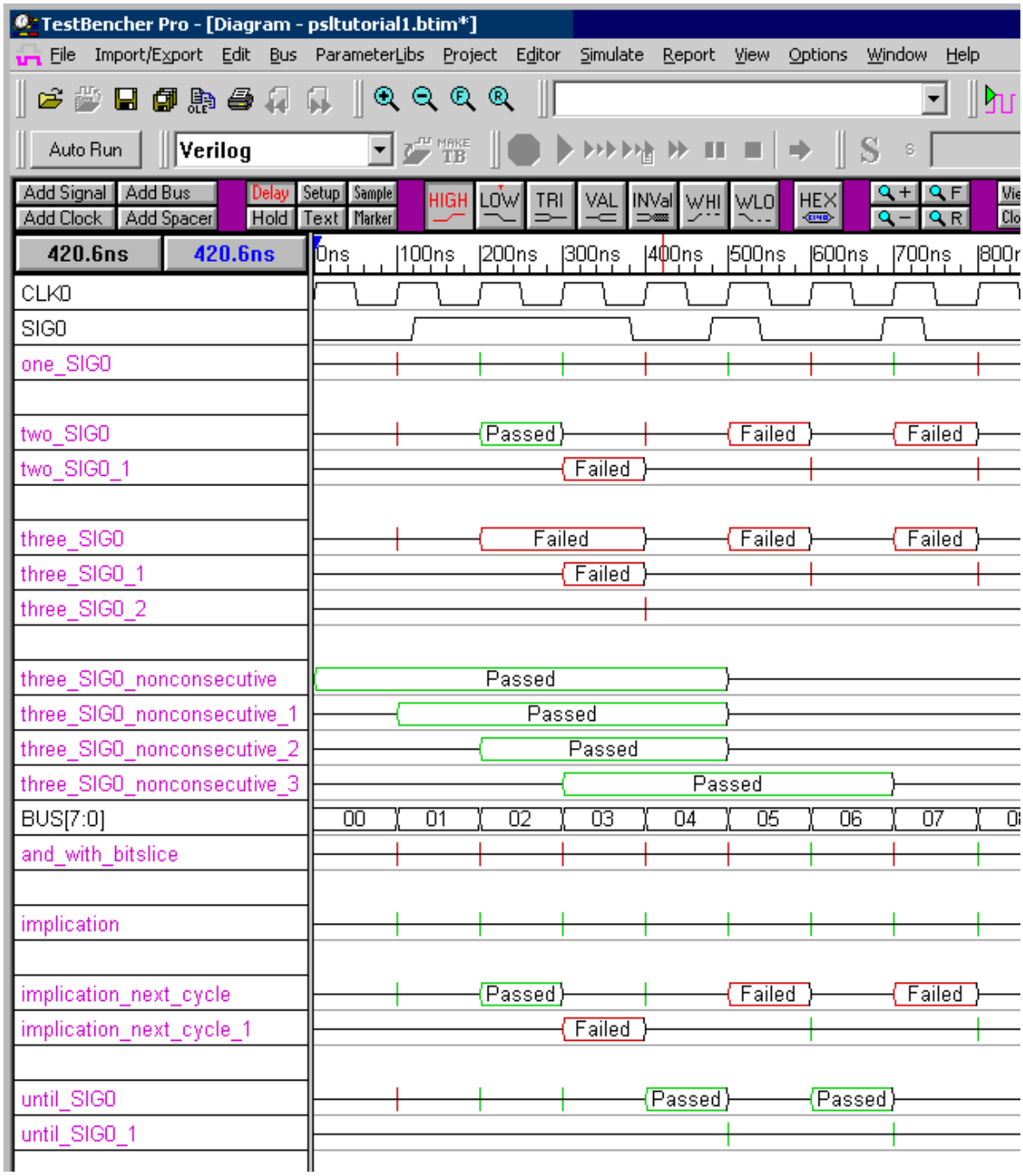
(SDC) 9: Summary

Congratulations on completing the SDC Timing Generation Tutorial. Any timing diagram can generate SDC commands. However, when starting with a new timing diagram specifically for generating SDC commands, it is usually a good idea to create all the clocks first. Next, work through each external device by adding the control and data signals and the associated timing parameters. Signals that are inputs to your design will usually have associated delays relative to the clock that can be obtained from external device datasheets (i.e. clock-to-output times). Similarly, outputs from your FPGA will need to meet the setup and hold times specified by external device datasheets.

Copyright © 2014, SynaptiCAD Sales, Inc.

Transaction Tracker Tutorial

This tutorial explores semantic differences between some of the most commonly used PSL operators. The assertions in this tutorial have been kept very simple, so that it is easy to see the differences between the operators. It is important to understand these distinctions before attempting to write practical, real world assertions.



For all the examples in this tutorial, we use two input signals (SIG0 and BUS[7:0]) and all the assertions are clocked off the positive edge of CLK0. The assertions will make a new match attempt on each positive edge of CLK0, even though previous match attempts by the assertion are still in

Copyright © 2014, SynaptiCAD Sales, Inc.

progress. When such overlapping match attempts occur, additional "overflow" signals will be created to display the resulting transaction records without overlapping them on a single result signal. These overlap signals are creating dynamically as needed during the simulation, and get cleaned up automatically at the beginning of any new simulation run.

The above image shows all of the results for the following equations:

- Match all occurrences of simple pattern 250 one_SIG0 = {SIG0}
- Match consecutive occurrences with Concatenation Operator 251 two_SIG0 = {SIG0;SIG0}
- Match with consecutive Repetition Operator 251 three_SIG0 = {SIG0[*3]}
- Match with non-consecutive Repetition Operator 252 three_SIG0_nonconsecutive = {SIG0[=3]}
- Bit-slices and the Boolean operators 252 and_ with_bitslice = {BUS[1:0] & BUS[3:2]}
- Implication operator 253 implication = {{SIG0} |-> {SIG0}}
- Implication Next-Cycle operator 253 implication_next_cycle = {{SIG0} |=> {SIG0}}
- PSL Property 253 until_SIG0 = ((BUS > 2) until SIG0)

(TT) 1: Open the Example File

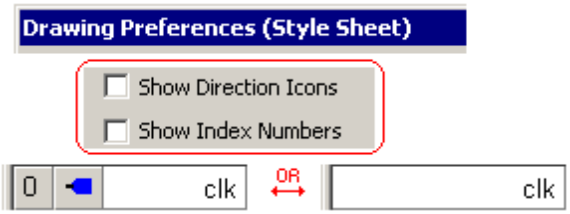
You will need a Transaction Tracker License to complete this tutorial. If you do not have one, please contact SynaptiCAD at sale@syncad.com.

- Choose the menu option **File > Open Timing Diagram** to launch the File dialog.
- Browse to the **c:\SynaptiCAD\Examples\sugar** directory and choose the **psltutorial.btim** file.

This file contains all the assertions discussed in this tutorial. As we go over each assertion, you can double click on the associated signal in the timing diagram to see how the assertion was specified.

Hide the Direction and Index Columns in the Label window:

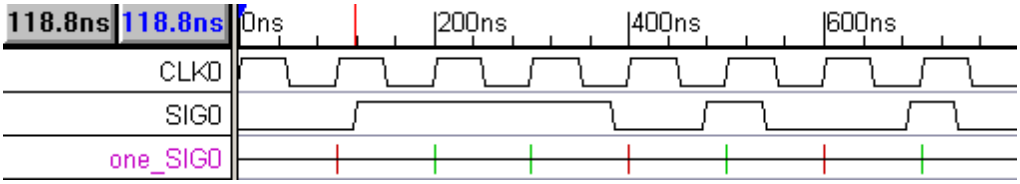
- Choose **Options > Drawing Preferences** to open the dialog. Then uncheck **Show Direction Icons** and **Show Index**.



(TT) 2: Match all occurrences of a simple pattern

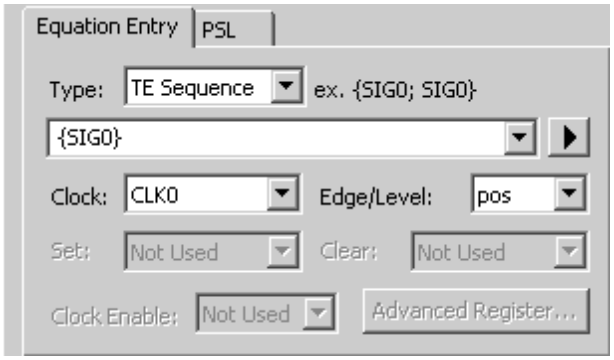
one_SIG0 = {SIG0}

This sequence matches a single occurrence of SIG0 on the positive edge of CLK0. Since it is a simple Boolean check, its transaction records have no time width and they display as spikes. It fails to match at time 100, so the first spike is red, then passes for two cycles because SIG0 is high, then fails again at time 400.

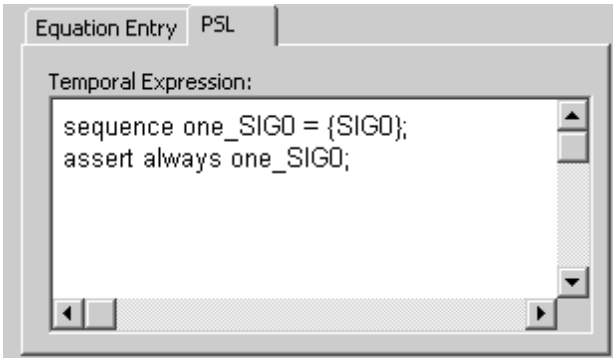


Copyright © 2014, SynaptiCAD Sales, Inc.

- Double click on **one_SIG0** to open the *Signal Properties* dialog. Note, that under the **Equation** tab, the **TE Sequence** is chosen. Also note that the equation is just **{SIG0}**.



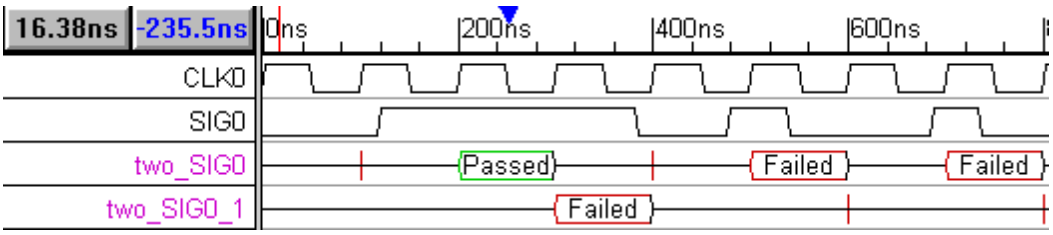
- If you click on the **PSL** tab, you can see the actual PSL code that is generated from the information in the TE Sequence equation and the other controls in the **Equation tab.**



(TT) 3: Match Consecutive Occurrences with Concatenation Operator

two_SIG0 = {SIG0;SIG0}

This sequence matches two consecutive occurrences of SIG0. The first attempt at time 100 fails immediately, because SIG0 is low, resulting in a red spike. The second attempt begins to match at time 200, and then succeeds at time 300, since SIG0 is still high at time 300. A new attempt is also begun at time 300, creating a transaction record that overlaps with the transaction record that started at time 200, so the time 300 transaction is placed on an overflow signal (two_SIG1_1). SIG0 goes low before time 400, so the time 300 transaction fails at time 400. A new transaction is also began at time 400, which fails immediately, resulting in a red spike at time 400 on two_SIG0.



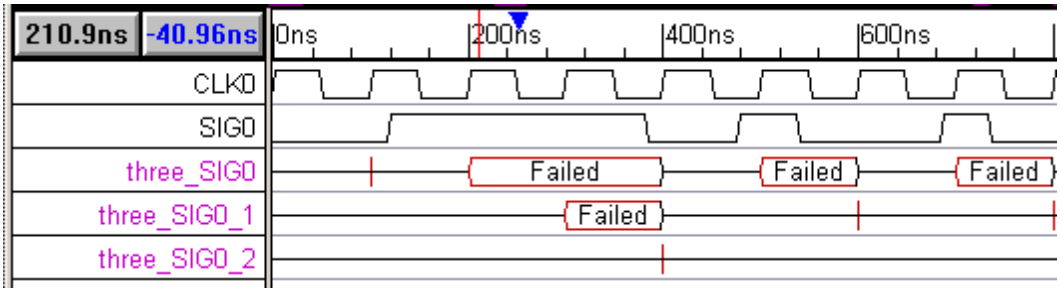
(TT) 4: Match with consecutive repetition Operator

three_SIG0 = {SIG0[*3]}

This sequences matches against 3 consecutive matches of SIG0. Note that this sequence could also have been specified as {SIG0;SIG0;SIG0}, but this gets awkward for specifying large numbers of repetitions of a pattern. There are no cases where SIG0 is high for 3 consecutive cycles, so all the match attempts eventually fail. The transaction record at time 200 holds for the first 2 cycles, then

Copyright © 2014, SynaptiCAD Sales, Inc.

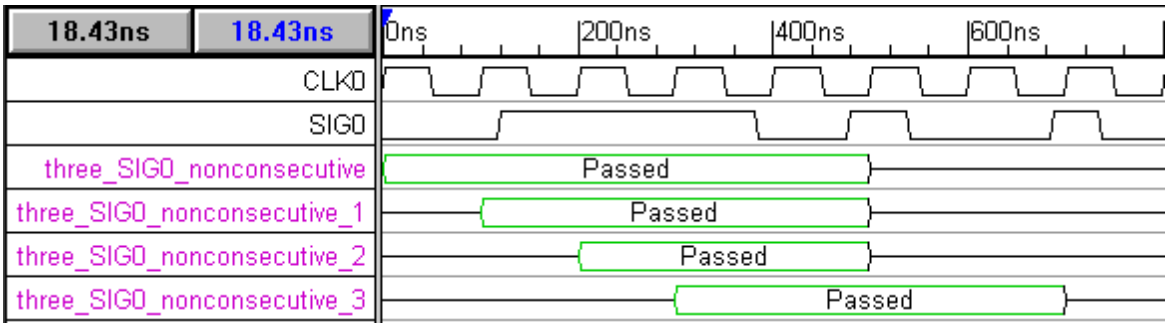
fails in the third cycle. Because it takes 3 clocks to fail, and two new matches are attempted during this time, two overflow signals are required to display all the transaction records without overlap.



(TT) 5: Match with non-consecutive Repetition Operator

three_SIG0_nonconsecutive = {SIG0[=3]}

This sequence looks for 3 **nonconsecutive** occurrences of SIG0 since we used [=n] instead of the consecutive operator [*n]. Note that SIG0 does not have to be true during the first cycle of the match attempt, this operator only that we eventually see 3 clocks during which SIG0 is true, so we end up with a large number of successful matches.

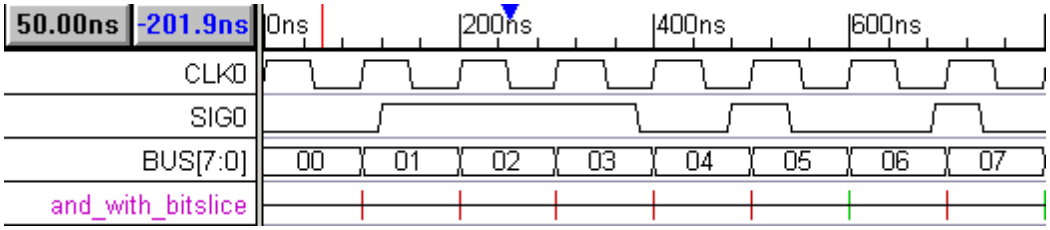


(TT) 6: Bit-slices and the Boolean operators

and_ with_bitslice = {BUS[1:0] & BUS[3:2]}

This assertion uses the **bit slice** operator to AND together the two least significant bits of BUS with the next two bits of BUS. The result of this operation is considered true if the value is greater than 0, so for Boolean truth, it is equivalent to: {(BUS[1] & BUS[3]) | (BUS[0] & BUS[2])}

The assertion matches at time 600 (when BUS has a value of 5, so BUS[3:2] ='b01 and BUS[1:0] ='b01) and at time 800 (when BUS has a value of 7, so BUS[3:2] ='b01 and BUS[1:0]='b11).



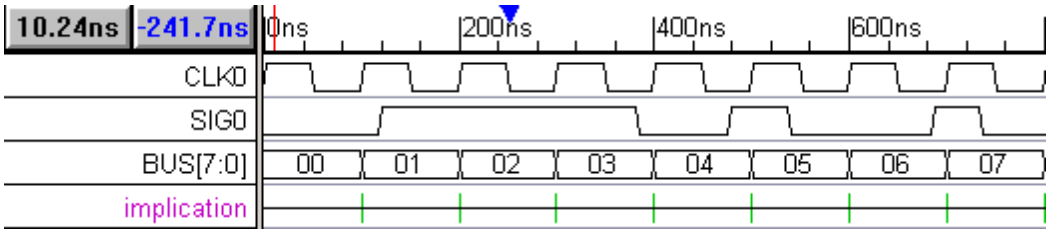
Copyright © 2014, SynaptiCAD Sales, Inc.

(TT) 7: Implication operator

implication = {{SIG0} |-> {SIG0}}

The implication operator guarantees that if the left hand operand holds (the sequence specified by the left hand operand succeeds), then the right hand operand will hold. If the left hand operand does not hold, the implication operator will succeed regardless of whether the right hand operand holds.

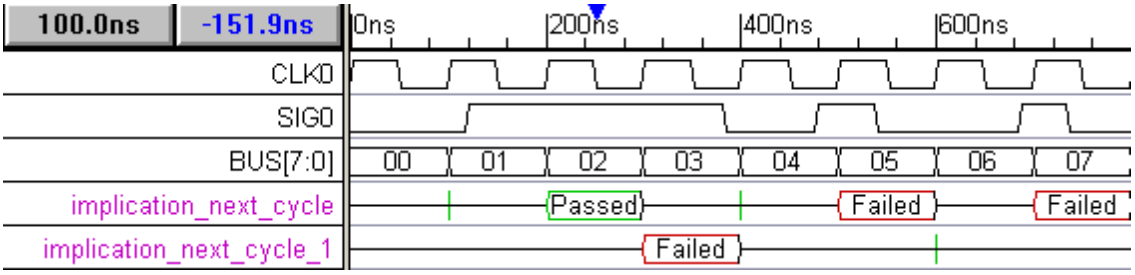
In this assertion we use the implication operator to ensure that: if SIG0 is true during a clock cycle, then SIG0 is true during this clock cycle. Therefore, this is not very a useful sequence as the sequence will always hold and matches immediately, regardless of whether SIG0 is true or false. Please note how this differs from the earlier {SIG0} assertion, which passes or fail each clock cycle based on whether SIG0 is true or not.



(TT) 8: Implication Next-Cycle operator

implication_next_cycle = {{SIG0} |=> {SIG0}}

This assertion looks very similar to the previous one, but the implication operator |-> has been replaced by the implication_next_cycle operator |=>. So this assertion is equivalent to: {{SIG0} |-> {true;SIG0}}. It checks that if SIG0 is true in the current cycle, it should be true in the next clock cycle. So if SIG0 is false at the start of a match attempt, that match attempt succeeds immediately. If SIG0 is true during the start of a match attempt, the match will succeed if SIG0 is true during the next cycle, or fail if SIG0 is false during the next cycle. Compare this to the operation of the earlier assertion {SIG0;SIG0} which fails when SIG0 is not true at the beginning of a match attempt.

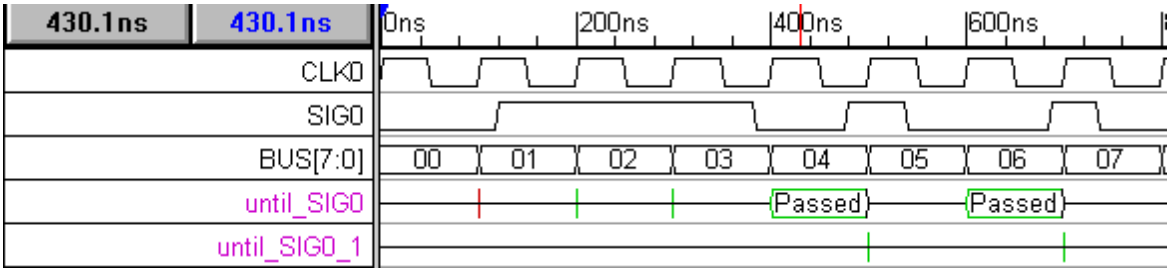


(TT) 9: PSL Property

until_SIG0 = ((BUS > 2) until SIG0)

For this signal we used a PSL property instead of an assertion, so the assertion body is surrounded by parenthesis instead of curly brackets. In the *Signal Properties* dialog, the equation edit box is **TE Property**. The property checks each cycle to see if the value of BUS is greater than 2 until SIG0 becomes true. Note that for the matches attempted at time 500 and 700, SIG0 is true on the initial clock cycle of the match, so the transaction record succeeds immediately.

Copyright © 2014, SynaptiCAD Sales, Inc.



(TT) 10: Summary of Transaction Tracker Tutorial

Congratulations, you have completed the Transaction Tracker Tutorial. You have viewed most of the basic PSL terms and seen how they return results. When writing your own PSL equations, try building up from smaller terms until you get the hang of the language.

Copyright © 2014, SynaptiCAD Sales, Inc.

Index

- A -

Adding 13, 15, 24, 25
clock 13
parameters 25
setups 24
signals 15
Analog 76, 77, 83
Ramps by a function 83
Ramps with mouse 76
Spice output 70
Step signals 77
step voltage export 70
Analog Display checkbox
piecewise linear 70
Analog Signals 70

- B -

Base Time Unit 12

- C -

Clocks 13, 24
adding 13
adding setup 24
Comparison 219
Masking Segments 219
Compile SynaptiCAD Library Models 192

- D -

Drawing Waveforms 15

- L -

Label Equation 83
Ramp 83

- M -

Masking Segments During Comparison 219
Match all occurrences of a simple pattern 250
Match Consecutive Occurrences with Concatenation Operator 251
Moving Signals 33

- O -

Open the Example File 250

- P -

Parameters 25
adding 25

- R -

Ramp function 83
Ramp Signals 70
Reordering Signals 33

- S -

Setups 24
adding 24
signals 15, 17, 33
adding 15
drawing waveforms 15
editing waveforms 17
moving 33
ramp 70
reordering 33
step 70
Step Signals 70

- W -

Waveform Comparison 219
masking segments 219
Waveforms 15
drawing 15
editing 17

Copyright © 2014, SynaptiCAD Sales, Inc.