DesignWorks™ Professional 5

Windows® Version

Simulator Option Users Guide

Revised for version 5.0
January 25, 2008

DRAFT

Some sections of this book have not been updated from the previous version. Any customer who has purchased the package and received this manual will receive the final version free of charge.

IMPORTANT NOTICE

Capilano Computing Systems Ltd. (“Capilano”) retains all ownership rights to the DesignWorks™ Professional program and all other software and documentation making up the DesignWorks package. Use of the DesignWorks software is governed by the license agreement accompanying the original media.

Your right to copy the DesignWorks software and this publication is limited by copyright law and your end user license agreement. Making copies, adaptations or compilation works (except copies for archival purposes or as an essential step in the utilization of the software) without prior written authorization of Capilano, is prohibited by law and constitutes a punishable violation of the law.

Capilano provides this publication “as is” without warranty of any kind, either express or implied, including but not limited to the implied warranties or conditions of merchantability or fitness for a particular purpose. In no event shall Capilano be liable for any loss of profits, loss of business, loss of use of data, interruption of business, or for indirect, special, incidental or consequential damages of any kind, even if Capilano has been advised of the possibility of such damages arising form any defect or error in this publication or in the DesignWorks software.

Capilano reserves the right to update this publication from time to time without notice. Some of the information in the publication refers to characteristics of third party products over which Capilano has no control. This information is provided for the convenience of DesignWorks users only and no warranty is made as to its correctness or timeliness.

Copyright ©1994-2004,2008 All rights reserved.

DesignWorks is a trademark of Capilano Computing Systems Ltd. Windows is a registered trademark of Microsoft Corporation. Other trademarks used in this publication are property of their respective owners.

Printed in Canada.

Capilano can be contacted at:

Capilano Computing
2631 Viking Way, Unit 218
Richmond, B.C., V6V 3B5
Canada

phone +1-604-522-6200

email info@capilano.com

WWW http://www.capilano.com

Table of Contents

Introduction 1

The DesignWorks Digital Simulation package consists of a number of tools that work interactively within the DesignWorks framework:

Simulatoris the digital simulation engine and is required for all simulation functions.

PLAprovides simulation models with PLA, PROM and RAM devices.

Timingdisplays timing diagrams.

Testused to display and run test vectors.

Chapter Organization

The DesignWorks manual is divided into the following chapters:

Chapter/Title	Comment
	Describes installation and initial startup.
	Describes simulation details including signals, devices, triggers and initialization.
	Describes schematic related issues including signal naming, power and ground, simulation models, probing as well as changes to the Schematic tool to support simulation
	Describes control of the simulator via the simulator window.
	Describes automatic or manual loading of simulation models.
	Describes displaying of timing waveforms.
	Describes the creation and execution of test vectors.

	Describes information on the simulation characteristics of the primitive device types.
	Describes the text format used for clipboard operations in the Timing tool.
	Describes the attribute fields used by the various simulation tools.
	Describes the available pin types and their effect on simulation.
	Describes pin orders and options for primitive devices.

Notes Regarding Copyright

The DesignWorks Simulator software and manual are copyrighted products. The software license entitles you to use the software on a single machine, with copies being made only for backup purposes, unless a specific license extension has been purchased. Any unauthorized copying of the program or documentation is subject to prosecution.

NOTE: Note regarding trademarks: A number of product trademarks are referred to in this manual. Full credit for these is given in the last section.

Tutorial–The Five-Minute Schematic and Simulation

2

This tutorial is divided into a number of sections, allowing you to review the basic functions first, then learn about more advanced features. The first section is entitled “The Five-Minute Schematic and Simulation” and will give you a taste of how quickly you can put together a circuit with full simulation. The later sections are divided by subject, so you can study in greater detail the features that are important for your application.

These tutorials are intended only to introduce you to DesignWorks features. For complete details on any subject, see the reference sections of this manual.

Tutorial Manual Format

In the following tutorial sections, text with a diamond:

like this

provides step-by-step instructions for achieving a specific goal. Other text provides background and explanation of the actions being taken.

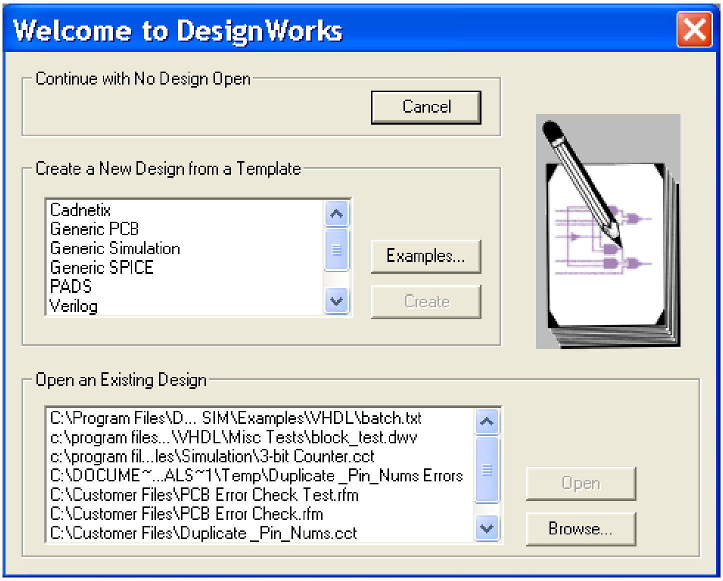
The Five-Minute Schematic and Simulation

In this section, we’re going to show how quickly you can create and test a circuit using DesignWorks.

Starting DesignWorks

Start the DesignWorks program by double-clicking on its icon.

Once the program has started, you will be looking at a welcome box like this.



Cadnetix
Generic PCB
Generic Simulation
Generic SPICE
PADS
Verilog

C:\Program Files\D... SIM\Examples\VHDL\batch.txt
c:\program files...\VHDL\Misc Tests\block_test.dwv
c:\program fil...les\Simulation\3-bit Counter.cct
C:\DOCUME~...ALS~1\Temp\Duplicate _Pin_Nums Errors
C:\Customer Files\PCB Error Check Test.rfm
C:\Customer Files\PCB Error Check.rfm
C:\Customer Files\Duplicate _Pin_Nums.cct

This box allows you to create a new design using a template, open a recently opened file, or select one of the example files included with the package. For our purposes, we will create a new, empty design.

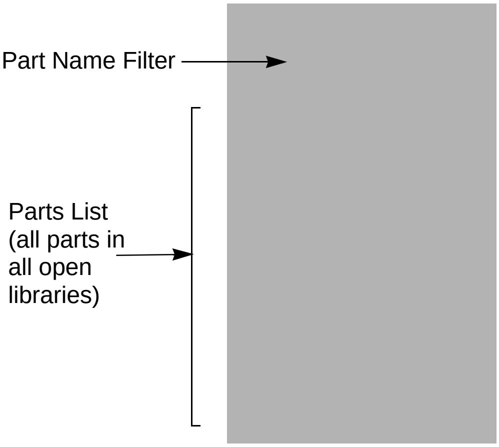
Double-click on the item "Generic Simulation" in the "Create a new design from a template" list.

You will now be looking at an empty circuit window like the following:

This window is your viewport onto the circuit diagram, which you will manipulate using the various drawing tools. The smaller panel at the bottom of the screen will be used by the program to display a timing diagram of the signals in your circuit, as well as other outputs generated by your circuit. Either of these windows can be moved or resized by the usual methods, to suit your needs.

Placing a Device

The parts palette shows a merged list of all parts in all open libraries. Libraries can be opened and closed manually using the Parts pop-up menu’s Open and Close commands, or any collection of libraries can be opened automatically at startup by placing them in the “Libs” directory.



The list of open libraries can be changed manually by doing either of the following:

Click on the File menu, select the Libraries submenu; then choose the New Lib or Open Lib commands, as needed.

Right-click on the Parts Palette; and use the same set of library commands that appear in that menu.

Any collection of libraries can be opened automatically at startup by modifying the INI file. This is described in more detail in the LogicWorks Reference Manual provided in electronic form with the software.

Locate the "Filter" text box on the Parts Palette. Type the text "164" into this box. This will reduce the parts list to only items containing that text.

Locate the part 74_164 in the parts list and double-click on it.

Move the cursor back into the circuit window. The cursor on the screen will now be replaced by a moving image of the selected symbol, in this case an 8-bit shift register.

The numbered devices in this library are generic 7400-series types. The labeling and simulation characteristics can be adjusted to match the various 7400 families on the market.

Position this image somewhere near the center of the circuit window and click the mouse button. A permanent image of the device will now stay behind in that location and the image will

continue to follow your movements.

More devices of the same type could be created at this point, but in this example we wish to select another symbol.

Press the spacebar to return to Point mode. Notice that you can click and drag the device that you placed to any desired new position.

Move again to the Parts Palette and this time double-click on the XNOR-2 type. (You might need to change the text in the “Filter” box, if you used it in a previous step.) Once you move outside of the Parts Palette, the cursor will immediately change to match the new symbol.

The XNOR-2, and the devices in the Simulation Gates, Simulation Logic, and Simulation I/O libraries, are called “primitive” types because they have built-in simulation models in DesignWorks. Other devices, such as those in the 7400 library are called “subcircuit” types because their simulation models are made up of primitives. If DesignWorks is being used only for schematic entry, it is also possible to make symbols with no simulation function.

Place one of these Exclusive-NOR gates adjacent to the 164 device so that the pins just touch, and click once to anchor the device.

Press the spacebar to return to Point mode.

Whenever you place devices or signal lines so that they touch, you will notice that the signal lines flash briefly. This indicates that a logical connection has been made. You do not need to explicitly request a connection.

Moving a Device

Point at the Exclusive NOR gate and click and drag to the right. While you hold the mouse button you can drag the device to any desired new position. Note that any signal lines attached to the device are adjusted continuously to maintain connection.

Position the gate as shown to the right of the 164 device.

Drawing Signal Connections

Attach a connection to the output of the gate by positioning the pointer near the endpoint of the pin and dragging away to the upper left.

Notice that two lines at right angles will follow your mouse movements to connect the starting and ending points.

While moving the mouse, try pressing the and/or keys and note the different line-routing methods available. Click mouse once to anchor the signal line.

For details on these line-routing modifier keys, see the section on Signal Line Editing in the LogicWorks Reference Manual provided in electronic form with the software.

Leave a right-angle line attached to the gate, as shown.

Extend this line to connect to the B input of the 164 by clicking at the line endpoint where you left off, dragging the line to the B input, and releasing the mouse button.

Add a connection to pin A by clicking at the end of the pin, dragging the line down until it touches the signal line, then releasing the mouse button.

Notice that an intersection dot appears automatically whenever three or more lines intersect.

Try repositioning a line segment by clicking and dragging anywhere along the length of the segment except at a corner or intersection.

Binary Switch Input Device

Return to the Parts Palette and select a Binary Switch device.

Place it as shown on the diagram.

Press the spacebar to return to Point mode.

Try clicking on the switch. Notice that it changes between the 0 and 1 states.

In order to move a switch, you must first select it by holding the key while clicking on it. This is necessary because the switch has a special response to a normal mouse click.

The devices in the Simulation I/O library can be used to actively control and observe the simulation right on the schematic. Each of these devices responds immediately to changes in the simulation in progress. The Hex Keyboard device is similar to the switch except that it operates on four lines at once.

Clock Generator Device

Select a Clock device and place it on the diagram just below the switch.

Press the spacebar to return to Point mode.

Route wires from the switch and clock to the 164, as shown. Remember to try using the and keys on the keyboard, to route the wires.

While you have been working on the diagram, the DesignWorks simulator has been running continuously, simulating the effects of the new connections that are being made. So far, though, we have not asked it to display any results. This is done either by placing probes on the diagram or by displaying signals in the Timing window.

Naming a Signal

> **Click on the text tool in the Tool Palette. The cursor will then change to a pencil shape, which will be used to select the item we want to name.**

Text Tool

The text cursor is used to name devices and signals, to apply pin numbers to device pins, and to add free text notations to the diagram.

> **Position the tip of the pencil anywhere along the length of the line running from the clock device, and click. A blinking insertion marker will appear.**

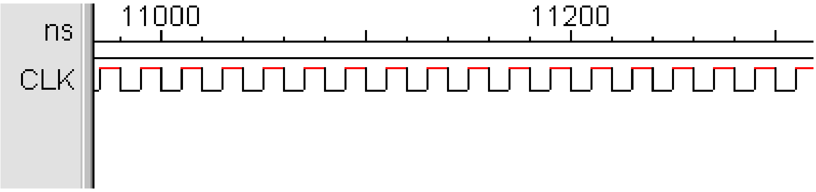
Type the name “CLK” on the keyboard, then press the key or click the mouse button once outside the text box.

Return to Point mode by clicking the arrow icon in the Tool Palette. Note that the name can be dragged to any desired position.

Click once on the Binary Switch to change it to the logical 1 state.

The Timing Window

You will immediately see the Timing window come to life with the displayed values on the CLK line. By default, any named signal is shown automatically in the Timing window, although you can change this using the Add Automatically command in the Simulation menu.



Again using the text cursor, name the two data lines from the shift register and the output line from the gate, as shown. The simulated output from these lines will immediately appear in the Timing window.

Simulation Controls

Click on the <> and >< buttons and observe that they affect the time scale of the Timing window.

Display resolution can be adjusted over a wide range of time values to suit the displayed data.

Select the Timing Window item in the View menu. You will notice that the Timing window disappears and the current time indicator in the Simulator palette advances much more quickly.

Select the Timing Window command again or use the corresponding button to re-enable the display.

Click on the Reset ( ) button and notice that the simulation restarts at time 0.

Adjust the speed slider control in the Simulator toolbar and notice that simulation slows.

Click repeatedly on the Step ( ) button and observe that the simulation proceeds one step at a time.

Click the Run button in the Simulator toolbar.

NOTE: The Step button advances the simulation to the next time at which there is some circuit activity, not necessarily just one time unit. The size of the step will depend upon the circuit.

Probe Device

Select the Binary Probe type from the Parts Palette.

Place a probe so that its pin contacts a signal line to view the simulation value on that line.

As the simulation progresses, the values on all probes are updated immediately. A similar device, the Hex Display, is also available to show groups of lines in hexadecimal. These simulation devices can be flagged to indicate that they are not a real part of the finished product and should not be included in any netlists or bills of materials.

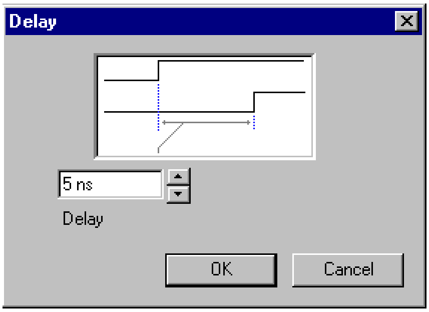
Setting Device Parameters

Click in the window, but away from any circuit objects. This deselects everything.

Click on the XNOR gate to select it.

Select the Simulation Params command in the Simulation menu.

Use the controls in this box to increase the propagation delay in the device to 5 ns, as shown.



Delay: 5 ns

The Simulation Params command is used to view and set delays associated with devices and pins. Pin delays normally default to zero but can be used to fine-tune the delays for different paths through a device.

Click on the OK button.

Device Delay on the Timing Window

Notice that the altered device delay immediately affects the simulation. You will see an increased delay between the clock reference lines and the changes in the FEEDBK signal.

Interacting with the Simulation

Try clicking on the switch hooked to the CLR input. Notice that it changes state and immediately affects the displayed simulation results.

Saving the Design

Click the Save button ( ), and save your circuit so you can continue with it later.

This ends the Five-Minute Schematic and Simulation tutorial section.

3

Tutorial–Structural Simulation

This section of the tutorial will provide you with a closer look at the integrated digital simulator in DesignWorks, including the following topics:

Types of devices simulated

Controlling the simulation

Representation of time and signal values

Using the trigger

Using the signal probe.

Logic States

Create a new circuit using the New command in the File menu.

Create the partial circuit below using the Buffer-1 O.C.and Binary Probe devices.

DesignWorks uses a total of 13 different logic values for signals in order to handle different drive levels and unknown situations. The probe will display an X for any of the six possible “Don’t Know” states. In this case, the X results from the fact that the device input is unconnected.

Add a Binary Switch device to the input of the buffer, as shown.

Click on the switch a couple of times and note that the buffer output alternates between the 0 and Z states.

The Z value indicates a high-impedance or undriven line. Multiple open-collector or three-state devices can drive a line to simulate bus or wired-AND logic.

Circuits with Feedback

Click on the buffer device and use the Duplicate command in the Edit Menu to create another one as shown.

Add a pullup resistor (using the Resistor and +5V symbols) and an inverter ("NOT" in the Simulation Gates library), and wire them as shown.

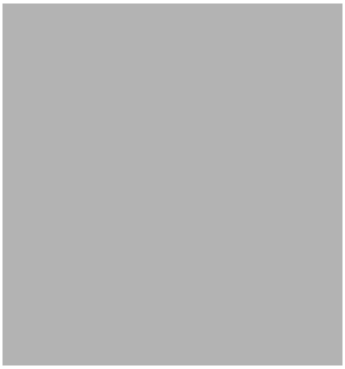
Click on the switch and notice the oscillation that occurs due to the feedback in this circuit.

Name the output signal CLK so that it shows in the Timing window.

Using the Signal Probe Tool

Click on the signal probe tool in the toolbar.

Click the tip of the probe tool along any signal line. It will show the current value of the signal as the simulation progresses.



You can also use this tool to enter new signal values by typing 0 or 1 on the keyboard while the mouse button is pressed. Stuck-at, unknown, and high-impedance levels can also be inserted.

Time Values

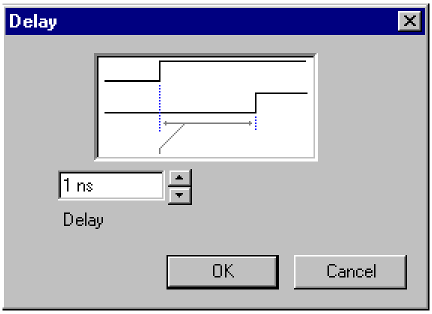
DesignWorks uses integers to represent simulated time values. The smallest unit of time is the femtosecond, written FS, and denoting 10^{-15} seconds. Most devices included with DesignWorks default to a delay of 1NS (nanosecond, or 10^{-9} seconds).

DesignWorks uses an *event-driven* simulator, meaning that device values are only recalculated when an input change occurs. Thus, the speed at which the simulation occurs does not depend on delay or other time values in the circuit.

Primitive Devices

Click on the inverter (NOT) device with the arrow tool to select it.:

Select the Simulation Params command in the Simulation menu.



1 ns

Delay

The inverter is classified as a *primitive* device since its simulation function is built into the program. Primitive devices have a single time value that defines the delay from any input pin to any output pin for any transition. More complex models can be implemented by using pin delays or by building subcircuit devices out of the existing primitives.

Click in the delay value box and change the number to 5 ns, then click on the OK button. Notice the effect this delay change has on the period of the oscillation in this circuit, as displayed in the timing diagram.

Power and Ground Signals

Select a 74_161 4-bit counter device from the 7400 library and place it in the circuit diagram as shown.

Using the text tool, as described previously, add the names "CLK" and "ENABLE".

Place +5V and Ground symbols as shown to permanently fix these signals to high and low levels, respectively.

Subcircuit Devices

Add the names D0 to D3 using the following procedure:

Name the least significant counter output D0 using the usual technique.

Hold down the key on the keyboard while you click on each higher pin in turn. Make sure you click only at the very end of the pin. This will automatically place sequential numbers on the lines clicked.

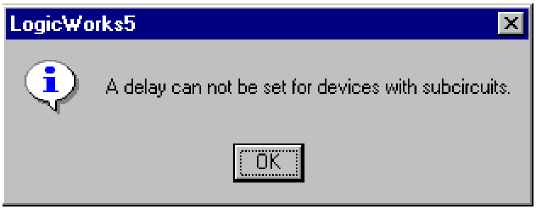
Notice that the traces D0 to D3 in the Timing window will show unknown values because the counter has never been cleared into a known state.

Press the (Clear X) button in the Simulator toolbar.

This resets all storage elements to the zero state and clears unknown lines.

Reactivate the arrow cursor.

Click on the 161 device to select it, then select the Sim Params command in the Simulation menu.



A delay can not be set for devices with subcircuits.

The 161 counter is a *subcircuit device*, meaning that its logic function is implemented using a combination of the DesignWorks primitive devices. Because of this, the overall delay for the device cannot be adjusted by simply changing one parameter. Two methods are available for modifying delays in subcircuit devices and these are discussed in the following sections.

Click the OK button on the warning box.

Right-click on the 161 device.

In the pop-up menu, select the Device Info command.

Click on the “Lock Opening Subcircuit” check box, to turn it off.

Click on the OK button to close the dialog.

Double-click on the 161 device to open its internal circuit.

A new window will open showing the internal circuit of this device. Notice how you can use the signal probe tool, the Parameters command, and all the drawing tools to view and modify this internal circuit. If you modify this circuit, *all devices of the same type* in this design will be equally affected.

Close the internal circuit by clicking in the X control at the top right corner of its window.

Pin Delays

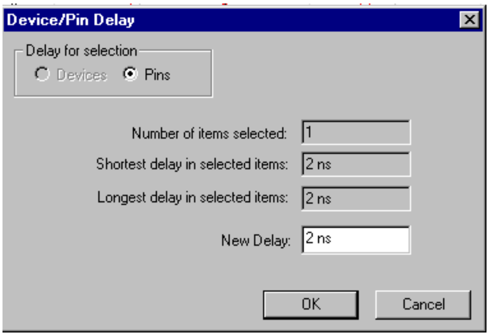
Using the arrow cursor, click midway along the QA output pin on the 161 device to select it.

Select the Simulation Params command from the Simulation

menu.

DesignWorks allows you to set a delay on an individual pin on a primitive or subcircuit device. The logical effect is the same as if you had inserted a buffer device with the specified delay in series with the pin. Pins always have a default delay of zero.

Set the pin delay to 2 NS and click OK. Notice the effect this has on the D0 trace in the Timing window.



Device/Pin Delay

Delay for selection
○ Devices ● Pins

Number of items selected: 1
Shortest delay in selected items: 2 ns
Longest delay in selected items: 2 ns
New Delay: 2 ns

Pin delays can be used to customize path delays in subcircuit devices without opening and modifying their internal delays. Setting pin delays on a subcircuit device affects only the single device modified, whereas changing internal primitive device delays will affect all copies of the same type of device.

Moving Timing Traces

Click and vertically drag the name CLK in the Timing window and reposition it relative to the other traces.

You can reposition any group of traces for ease in making timing comparisons. Any number of traces can be moved at once by holding the key while clicking on the trace names.

Grouping Timing Traces

Click on the name D0 in the Timing window. Hold the key depressed while you click on names D1, D2, and D3 so that they are all selected.

Press the right mouse button on any of the four selected names.

In the pop-up menu, select the Group command.

You will now see that the four traces D0 to D3 collapse into a single grouped trace showing their combined value in hexadecimal. The same pop-up menu can be used to Ungroup the signals again, or to set the signal order used to create the hexadecimal value.

Note that the grouped trace has double vertical bars on some values. This is due to the delay we inserted in the QA output pin. If you set the pin delay back to zero, this will disappear.

NOTE: The hexadecimal value of a grouped signal will only be displayed if there is sufficient space between the signal changes to display the text. You can use the ◇ ✕ (Zoom In/Out) buttons to change the scale factor to see the values.

Using the Trigger

Click on the Trigger button ( ) in the Simulator toolbar.

The trigger mechanism allows you to detect various timing and signal-state conditions.

Type the name CLK in the Names box.

Type the value 1 in the Value box.

Select the Reference Line option.

Click the OK button.

You will now see that a reference line is drawn on the Timing window each time the CLK signal changes to a 1 state. You can also enter ranges of signal names (e.g. D7..0) and corresponding hexadecimal values (e.g. 7A) into these boxes to match more complex events.

This completes the tutorial section on structural simulation.

Tutorial–Using VHDL in DesignWorks

4

In this tutorial section, we'll look at how you can create design descriptions and simulation models using the VHDL language. DesignWorks allows you to create designs containing a mix of structural components (that is, schematic diagrams) and VHDL. The following topics will get you started in creating each of these types of simulations and tying them together..

Creating a Simple VHDL Simulation

In this section, we're going to a simple, self-contained VHDL simulation from scratch.

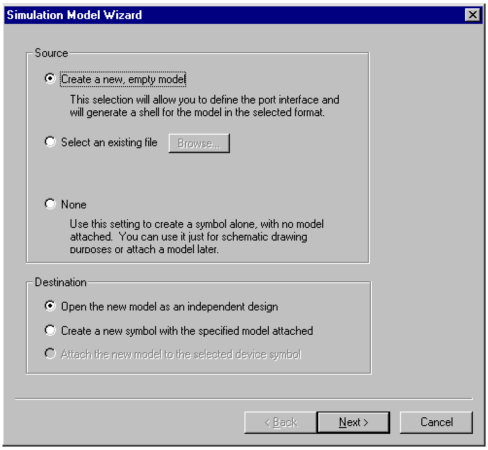
Creating a new VHDL Model

Go to the File menu and select the New command.

In the list of available document types, select Model Wizard and click OK.

The Model Wizard allows you to create either an independent, top-level design file or a component that can be used inside other designs. Any model can be created using either VHDL or a schematic circuit diagram.

The first panel of the Model Wizard looks like this:



Source

● Create a new, empty model

This selection will allow you to define the port interface and will generate a shell for the model in the selected format.

○ Select an existing file

○ None

Use this setting to create a symbol alone, with no model attached. You can use it just for schematic drawing purposes or attach a model later.

Destination

● Open the new model as an independent design

○ Create a new symbol with the specified model attached

○ Attach the new model to the selected device symbol

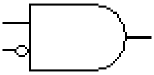
In the Source selections, choose “Create a new, empty model”.

In the Destination selections, choose “Open the new model as an independent design

With these selections, we are essentially creating a new, independent circuit. That is, it will not at this stage be used as a description of a component used in another design.

Click Next

This next pane allows you to choose which type of model you wish to create. In this case, we’re going to use VHDL create a simple AND gate with one inverted input that would look like this, if we made an equivalent logic diagram:



Select VHDL

Enter a name for the new model, such as AND1INV

NOTE: Since this name will be used in the VHDL source file, you cannot use a VHDL reserved keyword or anything containing invalid characters as a name. For example, AND would not be a valid name.

Click Next

We now specify the "interface" to the model, that is, what its inputs and outputs will be. In this case, we wish to add two single-bit inputs and one single-bit output. To do this:

Set the function to Input, if it is not already

Enter the name `POS` **for the first input. The note above about names in VHDL applies also to input and output names, so you have to be sure to use something that isn't a reserved word.**

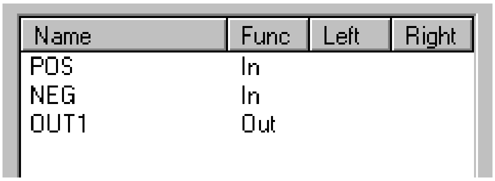
Click the Add Single Bit button.

Enter the name `NEG` **click Add Single Bit again to add the second input.**

Go back to the top of the panel and change the Function selection to Output.

Enter the name `OUT1` **and click Add Single Bit.**

The port list should now look like this:



Name	Func	Left	Right
POS	In		
NEG	In		
OUT1	Out		

IMPORTANT: The settings in the Func column must appear as shown!

Click the Finish button to create the model file.

You should now see a new document window open containing text like this:

```
library IEEE;
use IEEE.std_logic_1164.all;



entity AND1INV is


 port(
         POS : in        std_logic;
```

```
                NEG: in         std_logic;
                OUT1: out       std_logic
        );

end AND1INV;


architecture arch1 of AND1INV is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

This is now a complete VHDL description of a component having the desired inputs and outputs, except that no code has been added to describe the actual behavior of the device. Before we proceed, verify that this is a correct VHDL file.

Go to the VHDL menu and select the Compile command.

You will notice that a new panel appears at the bottom of the screen with the compilation results. You should receive a warning that output OUT1 has not been assigned.

Locate the line that starts "-- Your VHDL code" and replace it with the following signal assignment statement:

```
OUT1 <= POS AND NOT NEG AFTER 1NS;
```

Running the Simulation

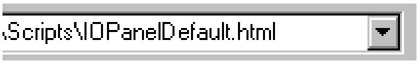
Click the (run) button to start the simulator.

You will now see the VHDL text document turn gray to indicate that it cannot be edited while the simulation is running. Now we need a method of feeding inputs into our design and checking the outputs.

Click the (I/O Panel) button. This will display a new panel in the results area at the bottom of the screen.

NOTE: If the I/O Panel has already been used, you might need to click the I/O Panel tab in the results panel in order to bring it to the top.

If it is not already displayed, click the selection list at the top of the I/O Panel and choose the item IOPanelDefault.htm



\Scripts\IOPanelDefault.html

The I/O Panel is actually a special kind of web page that can be programmed to display simulation results in many different ways. This default display shows the top-level signals in the design being simulated.

Try clicking the "0" controls to set the inputs to an initial zero state, then the + controls in the `pos` **and** `neg` **lines to change the input values to these two inputs. Note that the circuit obeys the appropriate truth table:**

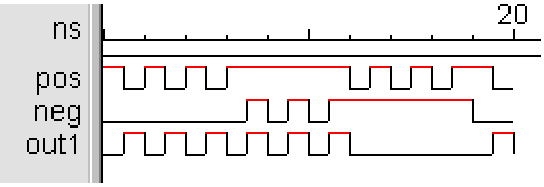
POS	NEG	OUT1
0	0	0
0	1	0
1	0	1
1	1	0

Displaying Timing Results

The I/O Panel is a quick way of viewing circuit inputs and outputs, but gives you no information about the relative timing of signal changes. To view the signals over timing, we will use the Timing window.

Click on the (Timing) button to display the timing diagram.

Since the timing diagram was actually collecting results while you were using the I/O Panel, it will display the changes that have occurred up to now.

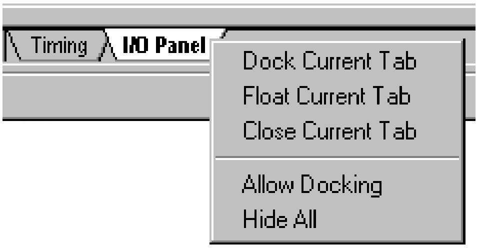


If necessary, you can use the ◇ ✕ (Zoom In/Out) buttons to adjust the resolution of the timing diagram to get a clear display.

Normally, results windows all share the same panel at the bottom of the screen. If you wish to view the Timing and I/O Panel windows at the same time,

Click on the I/O Panel tab to bring it to the front.

Click the RIGHT mouse button on the I/O Panel tab and select the “Float Current Tab” command.



This places the I/O Panel in a separate, floating window so that you can view both at the same time. You could also have done this to the Timing tab, if desired.

Now try changing the input values again and watch the effect in the timing diagram.

Note that there is a 1 nanosecond delay between input changes and the corresponding output change. This is due to the “AFTER 1 NS” specification in the VHDL model.

Creating a VHDL Model for a Device Symbol

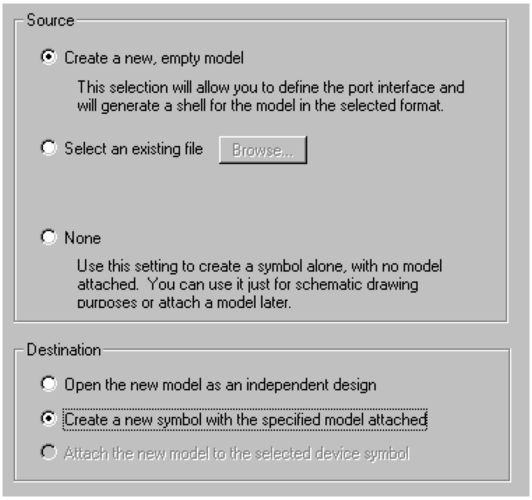
We'll now look at how we can use VHDL to describe the operation of a device that is going to be used in a DesignWorks circuit diagram. We'll also take the opportunity to use *vectors*, or multi-bit signals.

First, close any open circuit diagrams or VHDL files.

Select the New command in the File menu, then double-click on the Model Wizard selection.

For the Source selection, choose "Create a new, empty model".

For the Destination selection, choose "Create a new symbol with the specified model attached."



Source

● Create a new, empty model

This selection will allow you to define the port interface and will generate a shell for the model in the selected format.

○ Select an existing file

○ None

Use this setting to create a symbol alone, with no model attached. You can use it just for schematic drawing purposes or attach a model later.

Destination

○ Open the new model as an independent design

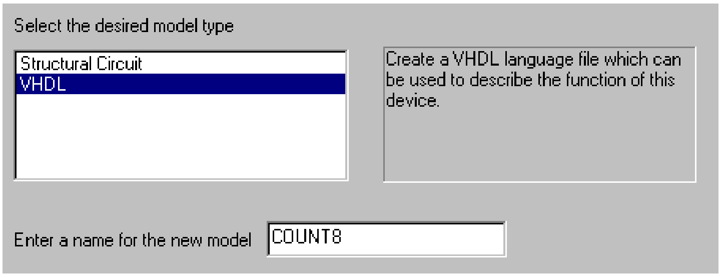
● Create a new symbol with the specified model attached

○ Attach the new model to the selected device symbol

Click Next

Select the VHDL model type and enter a name such as COUNT8

for the 8-bit counter device we are going to create.



Select the desired model type

Structural Circuit
VHDL

Create a VHDL language file which can be used to describe the function of this device.

Enter a name for the new model: COUNT8

Click Next to view the port interface panel.

Set the Function to Input, if it isn't already.

Enter the name `DIN` **into the name box.**

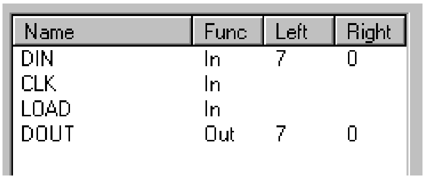
Enter the number 7 as the Left Bit Number and 0 for the Right Bit Number. Click the Add Vector button. This creates an 8-bit vector with bits numbered from 7 down to 0.

Change the name to `CLK` **and click Add Single Bit.**

Change the name to `LOAD` **and click Add Single Bit.**

Change the name to `DOUT`**, set the Function to Output and click Add Vector.**

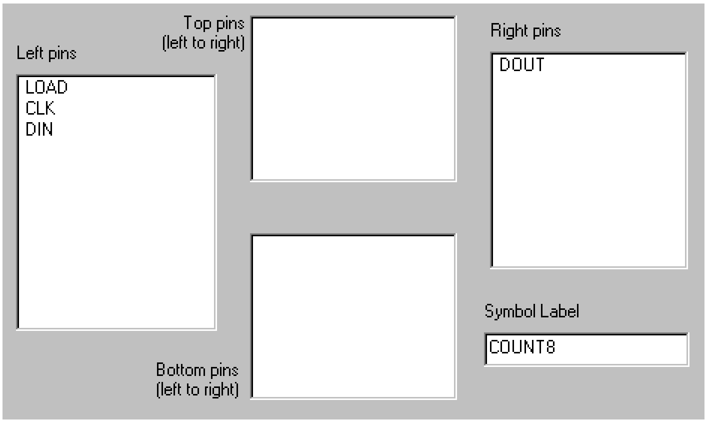
You should now see a port list like this:



Name	Func	Left	Right
DIN	In	7	0
CLK	In		
LOAD	In		
DOUT	Out	7	0

Verify that all the port settings are correct and click Next.

This next panel allows you to specify where the pins will appear on the schematic symbol. By default, inputs will be placed on the left and outputs on the right, which should make sense for most applications.



Left pins: LOAD, CLK, DIN

Right pins: DOUT

Symbol Label: COUNT8

If desired, move pins to different locations on the symbol by dragging and dropping names from one box to another.

Once you are satisfied with the pin locations, click Next.

This last panel allows you to decide which library you want to save the new symbol into.

If you already have a work library open in the list, you can select it now.

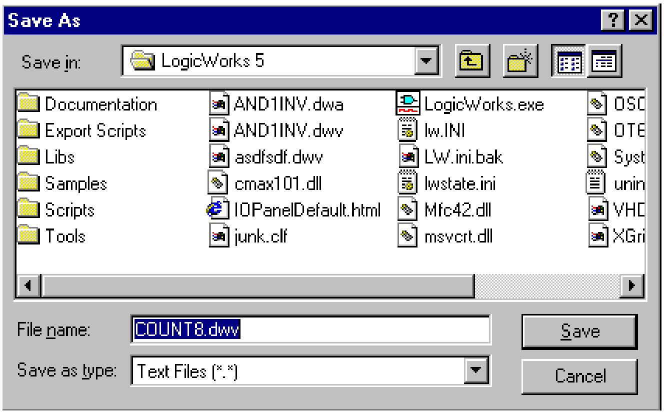
NOTE: We do not recommend saving your own components into the libraries supplied with DesignWorks. Future upgrades to the software might replace those libraries and you could lose your work.

If you do not have a work library open, use the Open Lib button to open an existing library, or the New Lib button to create a new one.

Once you have selected a library, click the Finish button. A standard Save As box will appear asking you to save the VHDL model file. This is necessary because the name of the file will be stored with the component.

Save the COUNT8 model file in the default location, or find any

suitable folder for it.



Save As

Save in: LogicWorks 5

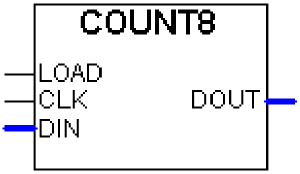
File name: COUNT8.dwv

Save as type: Text Files (*.*)

The Model Wizard has now created a VHDL model file that describes all the inputs and outputs, but has no actual behavior. It has also created a device symbol with entries linking it to the file. We can now have two steps left, first to fill in the actual behavioral part of the VHDL model, then to build a test circuit to make sure it works.

Select the New command in the File menu, then choose the Circuit item. This will create a new, empty circuit window on the screen.

Locate the COUNT8 part in the parts list on the right hand side of the screen and place one of these in the circuit. In case you're not familiar with this procedure, refer back to "Placing a Device" on page 22. You should now be looking at a circuit like this:



Double-click on the COUNT8 device. This will open the VHDL model in a new window.

After the line "use IEEE.std_logic_1164.all;" insert the following addition USE statement:

```
use IEEE.numeric_std.all;
```

This line is needed because we will be using some arithmetic data types and operations that are defined in this package.

Locate the line near the end of the file that says something like "-- Your VHDL code defining the model goes here:". (Remember that "--" indicates a comment in VHDL.) We are going to replace this line with some code that defines the operation of an 8-bit counter, as follows:

```
clk_proc : process(CLK)
     variable COUNT : unsigned(7 downto 0) := "00000000";
 begin
     if CLK'EVENT AND CLK = '1' then
      if LOAD = '1' then
          COUNT := DIN;
      else COUNT := COUNT + 1;
      end if;
     end if;
     DOUT <= COUNT after 500ps;
 end process clk_proc;
```

NOTE: Take care when entering the fourth line above. The item "CLK'EVENT" consists of the name `CLK` followed by an apostrophe (single quote) followed by the word `EVENT`. For more information on this VHDL attribute, see "Positive Edge-Triggered D Flip-Flop" on page 153.

Just to make sure we haven't made any errors, go to the VHDL menu and select the Compile command. You should get a message in the VHDL console window at the bottom of the screen indicating that the file compiled without errors. If any errors come up, fix them up before proceeding. Here are some things you might want to check:

The input and output names (such as `CLK`), must exactly match the declarations at the top of the file. If you didn't enter the names exactly the same way in the Wizard, it won't compile correctly.

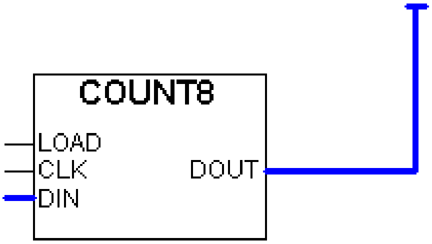
VHDL is very fussy about the positions of punctuation such as quote marks and semicolons.

"=", "<=", and ":=" can all be read as "equals" to us, but they have very different meanings!

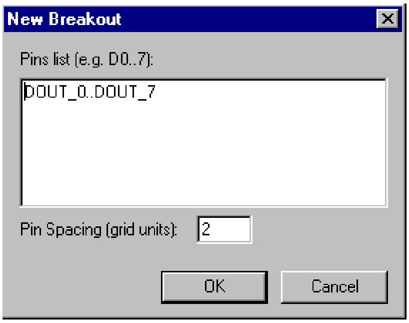
Once the file compiles correctly, close the COUNT8 document window. You should now be again looking at the circuit containing the COUNT8 symbol. The system description is now complete, so we just have to test it.

Although we could use the I/O Panel as we did in the previous tutorial, we'll take a different approach this time and add circuitry to the diagram to test the new device.

Click and hold at the end of the `DOUT` pin and extend the bus line as shown:



Right-click anywhere along the bus line and select the Breakout command. This will display the following box:



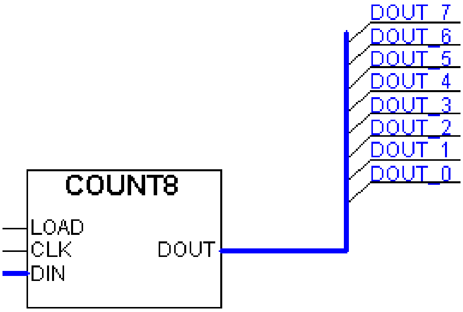
Pins list (e.g. D0..7): DOUT_0..DOUT_7

Pin Spacing (grid units): 2

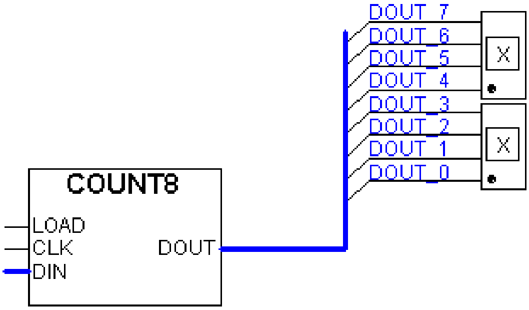
This box should already show the signal range "DOUT_0..DOUT_7", which means that all the individual bits from 0 to 7 will be split out of the bus. Click OK.

Place the breakout symbol so that it connects to the `DOUT` bus as

shown here:



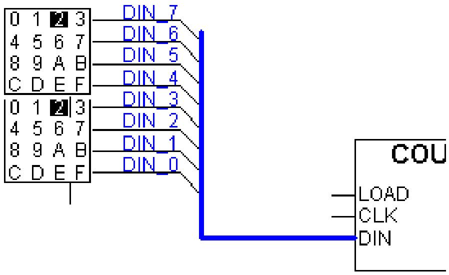
Next, locate the device Hex Display in the parts list and place one so that it connects to the bottom 4 pins on the breakout. Repeat this to place a second Hex Display to the top 4 pins.



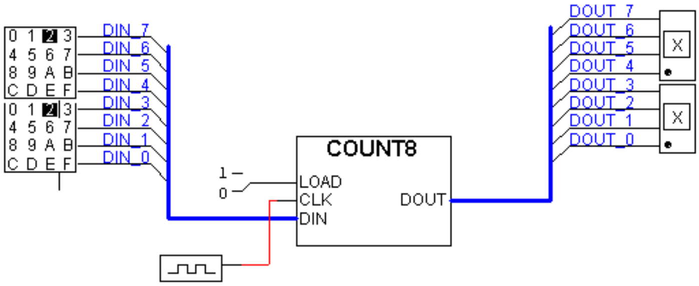
Using a similar procedure, extend the `DIN` **bus to the left, right-click on it to select the Breakout command, and place the breakout. You may need to use the arrow keys on the keyboard to orient the symbol as shown.**

Locate the device Hex Keyboard in the parts list. Double-click on it and move into the circuit area. You may need to use the arrow keys on the keyboard to orient it the right way around to attach to the breakout. Place two keyboards so they connect to the input

pins, as shown.



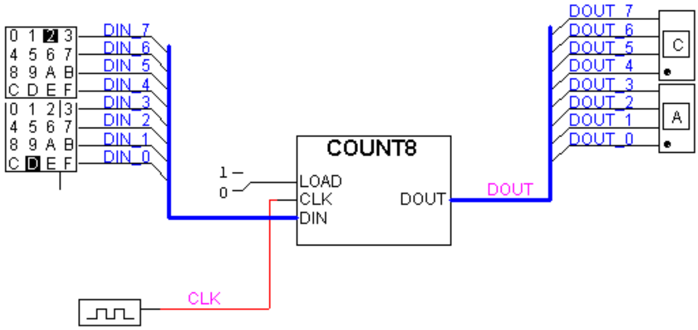
Place a Binary Switch device and wire it to the `LOAD` **input, then place a CLOCK device and connect it to the** `CLK` **pin. You should now have something like:**



To make it easier to display the results, lets apply names to a couple of the signals that we will want to observe. Select the

A (Text) too, then click on the bus line coming out of the `DOUT` **pin. Enter the name** `DOUT` **and press Enter on the keyboard. Use the same procedure to apply the name** `CLK` **to the output of the Clock device. You may want to move some of the devices around**

to make room for the names. Here is our final circuit:



This would be a good time to save this design, in case we want to come back to it later. Select the Save As command in the File menu and save it as "COUNT8 test" or any other name that suits you.

If it is not checked already, select the Show Values command in the Simulation menu, to show the values of signal on the circuit diagram.

Click the (Run) button to start the simulator.

You should now see the signal value displays change and the time indicator in the toolbar start to advance. Time is advancing because of the Clock device we placed in the circuit. This device generates a continuous sequence of 0/1 value changes at its output, regardless of what else is happening in the circuit.

Click on the LOAD switch to change its value to 1.

Click on the hex keypads at the inputs and observe that the input values are being transferred to the output on the clock.

If it is not already displayed, show the timing diagram by clicking on the (Show Timing) button or by clicking on its tab in the results window. If desired you can change the resolution of the timing diagram by using the (Zoom In/Out) tools.

Using a DesignWorks Symbol in a VHDL Design

In this tutorial, we will create a design using VHDL to create the top level description and having it refer to DesignWorks symbols as building blocks. This is the reverse of the situation described in the previous tutorial.

IMPORTANT: The VHDL language has more severe restrictions on names than the general DesignWorks program. In order for a symbol to be usable as a device model within a VHDL description, the name of the library itself, the name of the symbol and the names of all pins on the symbol must meet VHDL naming requirements. In general, this means that names cannot contain any spaces or special characters. Most of the libraries provided with DesignWorks do not meet these requirements, so you must either use the specific libraries provided for this purpose, or create your own versions of libraries that have appropriate names.

Select the New command in the File menu and double-click on the Model Wizard item.

In the Source selections, choose “Create a new, empty model”.

In the Destination selections, choose “Open the new model as an independent design

Click Next

Select the VHDL model type.

Enter the name FULL_ADDER for the new model

Click Next

We now specify the “interface” to the model, that is, what its inputs and outputs will be. In this case, we wish to add three single-bit inputs and two single-bit outputs. To do this:

Set the function to Input, if it is not already

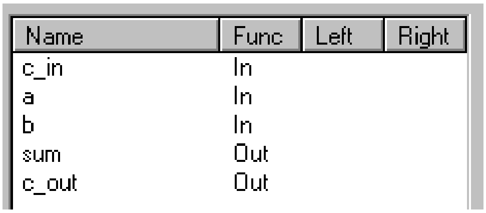
Enter the name “c_in” for the first input and click the Add Single Bit button.

Repeat the above procedure for inputs named “a” and “b”.

Go back to the top of the panel and change the Function selection to Output.

Add output bits “sum” and “c_out”.

The port list should now look like this:



Name	Func	Left	Right
c_in	In		
a	In		
b	In		
sum	Out		
c_out	Out		

Click the Finish button to create the model file.

You should now see a new document window open containing text like this:

```
library IEEE;
use IEEE.std_logic_1164.all;



entity full_adder is


 port(
          c_in  : in   std_logic;
          a     : in   std_logic;
          b     : in   std_logic;
          sum   : out  std_logic;
          c_out: out  std_logic
     );

end full_adder;



architecture arch1 of full_adder is


begin


  -- Your VHDL code defining the model goes here


end arch1;
```

To make this into a complete description, we will add component instantiation statements that refer to DesignWorks gate symbols stored in libraries.

In the VHDL file, locate the "use" statement close to the top of the file. After this line, insert the following additional statements:

```
library Libs;
use Libs.VHDLPrims.all;
```

These statements tell VHDL where to find the components that we will be referring to. The name, in this case "VHDLPrims", must refer to a library that is already open in the DesignWorks parts palette.

Locate the "architecture" statement in the file. On the next line, insert the following declaration:

```
signal s1, s2, s3 : std_logic;
```

This creates some intermediate signals that will be part of our model.

Find the comment line starting "-- Your VHDL code" that appears near the end of the file. Replace this comment with these lines:

```
G1 : xor_3 port map(INA => c_in, INB => a, INC => b, Y => sum);
G2 : and_2 port map(INA => c_in, INB => a, Y => s1);
G3 : and_2 port map(INA => a, INB => b, Y => s2);
G4 : and_2 port map(INA => c_in, INB => b, Y => s3);
G5 : or_3 port map(INA => s1, INB => s2, INC => s3, Y => c_out);
```

The "xor_3", "and_2" and "or_3" components are all items that will be fetched from the DesignWorks library "VHDLPrims". The component names and pin names must exactly match those defined on the symbol.

Use the Save or Save As command to save your file for safekeeping.

Click the (Run) button to start the simulator. If any compilation errors occur, they will be reported in VHDL console window, otherwise the VHDL window will go gray to indicate that it is locked while the simulation is running.

Click the (I/O Panel) button. This will display a new panel in the results area at the bottom of the screen.

Try entering values for the a, b, and c_in inputs and verify that

the model is working as expected. Here is the truth table for a full adder that our model should follow:

a	b	c_in	sum	c_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

NOTE: The delay values used by the DesignWorks symbols are determined by settings that were applied when the symbols were created. There is no way of changing the delays in the symbols themselves from the VHDL file, although you could apply additional delays to the signals using normal VHDL statements.

Simulation Overview 5

This chapter describes simulation details including signals, devices, triggers and initialization.

General Information on Simulation

DesignWorks has the ability to perform a realistic simulation of any digital circuit. Obviously, though, any simulation of any system must be limited in detail and must make certain assumptions. In particular, when simulating digital circuits, it must be understood that real circuits are never completely "digital" in nature, and in fact have many "analog" properties which affect how they operate.

DesignWorks is primarily intended to assist with the logical design of a circuit, and does not take into account factors such as line loading, power supply noise, rise and fall times, output drive, etc. As more of these factors are taken into account, the simulation becomes slower and less interactive, which defeats the purpose for which the DesignWorks simulator was created.

Type of Simulation

DesignWorks performs a discrete simulation of the signal changes in a logic circuit, meaning that signal levels and time change only in steps, rather than continuously. The program does not attempt to analyze your circuit, but simply tracks signal level changes through the devices. Thus, circuits with feeback loops or other delay-dependent features will be simulated correctly as long as they don't rely on particular analog characteristics of devices.

The simulation is "event driven", an event being a change in level of a signal. Each time an event occurs, a list is made of all the devices whose inputs are affected by that event. Any other events occurring at the same time are similarly evaluated, and affected devices added to the list. A type-specific routine is then called for each device on the change list in order to determine what output changes are going to occur. These changes are added to the event list, their time of occurrence depending upon the device delay. No computation is performed for times when no event occurs, so that device delay settings and clock values have no effect on how fast the simulation is performed.

DesignWorks performs strictly a digital simulation. It does not take into account factors such as fan-out (i.e. the number of inputs connected to a given output), line length (capacitance), asymmetrical output drive, etc., except in as much as these affect delay time.

Simulation Memory Usage

When a circuit is opened or created by DesignWorks, the circuit data is retained completely in the memory of your machine. Since the total memory available is fixed (until you buy your next memory expansion!), this places some limits on circuit size and simulation.

Each time a signal changes state, an "event" record is created in memory. If the signal is not being displayed in the Timing tool, then this record is deallocated again after the signal change has occurred. If the signal is being displayed, then the record is retained in memory until that change has scrolled off the left-hand side of the Timing window. As a result, the memory used by event records will increase when the number of displayed signals is increased, the resolution of the timing display is decreased or the "retain time" setting is increased.

Time Units

DesignWorks uses 32-bit signed integer arithmetic to calculate all time values used in the simulation. It is usually convenient to think of these values as being in nanoseconds, but the actual interpretation is left up to the user.

The simulation will wrap if any time value approaches the 32-bit integer limit.

Signal Simulation Characteristics

Signal States

DesignWorks uses thirteen different device output states in order to track conditions within your circuit. These states can be broken into three groups, as follows.

Forcing States (denoted by suffix .F):

LOW.F

HIGH.F

DONT01.F

DONT0Z.F

DONT1Z.F

CONF.F

Resistive States (denoted by suffix .R):

LOW.R

HIGH.R

DONT01.R

DONT0Z.R

DONT1Z.R

CONF.R

High Impedance:

HIGHZ

Note that the Forcing/Resistive distinction is used only to resolve conflicts between multiple outputs connected to the same signal. The final value stored or displayed for a given signal line can only be one of five possibilities:

LOW

HIGH

DONT

CONF

HIGHZ

Description of States

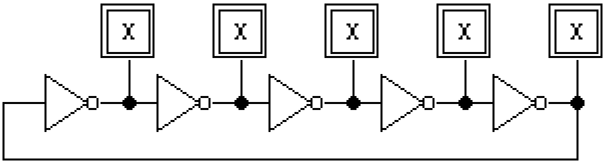
The High and Low states are the normal ones expected in a binary circuit, but are not sufficient to realistically simulate circuit operation, so the High Impedance, Don't Know and Conflict states are added. There will always be some cases where the simulation will not correctly mimic what would appear in a real circuit, and some of these cases are discussed below. In particular, if a circuit takes advantage of some analog property of a specific device, such as inputs that float high, known state at power-up, input hysteresis, etc., it will in general not simulate correctly.

High Impedance

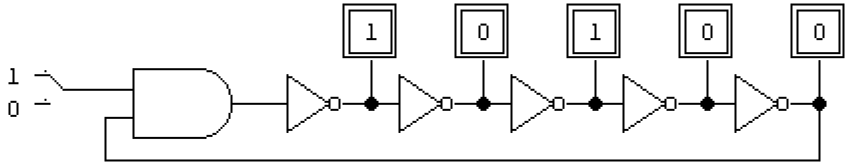
This state ("Z" on a binary probe) is used for cases when no device output is driving a given signal line. This may occur for an unconnected input, or for a disabled "three-state" or Open-collector" type device. If a device input is in the High Impedance state, it is treated as unknown for the purposes of simulation, even though in a real circuit the device may assume a high or low state, depending on the circuit technology used.

Don't Know

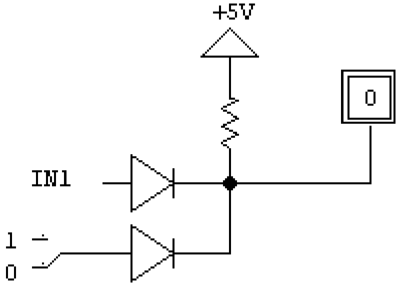
The Don't Know state is used in cases where the actual result in a real circuit would depend in the circuit technology used, on random chance, or on analog properties of the device not predictable using a strictly digital simulation. For example, if the following ring oscillator circuit is created in DesignWorks, all signals will be permanently unknown, since each depends on the previous one, which is also unknown. In actual hardware, this circuit may oscillate, or may settle into an intermediate logic level, which would not be defined in a digital circuit.



For the purposes of simulation, all circuits must have some provision for initialization to a known state. In most cases, circuits can be initialized using the Clear Unknowns command or by setting the initial value attribute, both described elsewhere in the manual. Alternatively, circuitry can be added to allow a reset to be done, as in the following modification to the ring oscillator:



A problem arises in simulating circuits with multiple open collector devices, such as a bus line, illustrated below:



In this circuit the upper device has an unconnected inputs at IN1 and therefore outputs a “Don’t Know” value. The lower device has a low input and therefore outputs a low value. In order to correctly resolve this situation the simulator needs to distinguish between a “Don’t Know” output from a normal “totem-pole” type output and a “Don’t Know” from an open collector, open-drain, or other single-drive output. In this case the upper device will produce a DONT0Z output which resolves correctly to a LOW on the output regardless of the state of IN1, using the rules described above.

Conflict

The "Conflict" state ("C" on a binary probe) results when two device outputs are connected and are of different or unknown state taking into account the rules described above.

State Display

The Timing tool displays the various signal states in different colors.

Stuck-at Levels

The DesignWorks simulator implements "stuck-at" levels to assist in setting initial simulation states, testing for faults, etc. When a signal is in a "stuck" state, it will not change state, regardless of changes in devices driving the line.

When the stuck status is set, the signal will retain the value it had at the time until some user action forces a change. When the stuck status is removed, the signal will return to the value determined by the devices driving the line.

Setting Stuck Levels

A signal can be placed in a Stuck High or Stuck Low state by any of the following means:

applying the name "0" or "1" to the signal.

typing "H" or "L" while viewing the signal value with the probe tool.

using the Stick High or Stick Low control in the Simulators Stick Signals dialog.

setting a "stuck" option in the Test Vector or other external simulation tool.

Each of these methods is described in more detail in the relevant sections of this manual.

Clearing Stuck Levels

The stuck status can only be cleared by one of the following user actions:

typing the spacebar while viewing the signal using the probe tool.

clearing the "stuck" state using the Simulators Stick Signals.

clearing the "stuck" state using the Test Vector or other external simulation tool.

Resolution of Multiple Device Outputs

The DONT0Z and DONT1Z values are used primarily to handle cases of open collector or open emitter devices with unknown inputs (see more information below). Most other types of devices produce the DONT01 output when a value cannot be calculated.

In cases where two or more device outputs are connected together and each one drives the line with a different value, the following rules are used to resolve the actual value on the line.

the forcing/resistive distinction is only used to resolve outputs from multiple devices. The final value used for display and simulation purposes is one of the forcing values or HIGHZ.

a forcing drive always overrides a resistive drive or HIGHZ (i.e. the signal takes on the value of the forcing drive, ignoring all resistive drives and HIGHZs).

a resistive drive always overrides HIGHZ.

DONT0Z.F and LOW.F produce LOW.

DONT1Z.F and HIGH.F produce HIGH.

any other combination of conflicting forcing drives produces CONF.

DONT0Z.R and LOW.R produce LOW.

DONT1Z.R and HIGH.R produce HIGH.

any other combination of conflicting resistive drives produces CONF.

Resistive vs. Forcing Drive

All primitive devices in DesignWorks output a forcing drive level except for the Resistor primitive device. The function of the Resistor device is to convert a forcing drive on one side into a resistive drive on the other. This can be used to modify the output of any existing device type by placing a resistor in series with it. Note that DesignWorks does not model analog properties of devices, so the resistor does not have a resistance value in the analog sense. In particular, there is no interaction between resistor and capacitor symbols to produce delay in lines. The delay effect can be simulated by setting a delay value for the resistor.

Probe Tool

The Probe tool is used to interactively examine and change values on individual signals and pins in the circuit schematic. When the probe tip is clicked and held on a signal or pin, the cursor will show the current value on the signal or pin and track changes that occur as the simulation progresses.

See "Probing a Signal" on page 87 for more information.

Busses

Busses (i.e. groups of signals represented by a single line on the schematic) have no particular significance to the simulator. The value of a bus is completely determined by the values of the individual signals it contains. The simulator performs no operations on the bus itself.

NOTE: NOTE: A bus can be displayed in the Timing window using the Add To Timing command, but this is really equivalent to displaying all the internal signals individually and then grouping them.

Bus Pins

Bus pins, like busses, have no particular significance to the simulator. The value of a bus is completely determined by the values of the individual pins it contains. The simulator performs no operations on the bus pin itself. Bus pins are not supported on primitive device types.

Device Simulation Characteristics

Device and Pin Delay

Primitive Device Delay

Primitive devices (i.e. those with a program-defined simulation model) have a single delay value which can be set to any integer value from 0 to 32767. This delay is applied when any input change causes any output change. In addition, a pin delay in the range 0 to 32767 can be set on any input or output pin. Pin delays can be used to set arbitrary path delays through the device. See more information on pin delays below.

The initial delay value is set to 1 when the device is created, but this can be changed later using the Parameters command in the Options menu in the Simulator window. This delay applies whenever any input change causes an output change. There is no provision in the built-in simulation models for different delay values on low-to-high and high-to-low transitions. The Clock and I/O devices have no delay characteristic.

See "Working With Hierarchical Blocks" on page 82 for more information on settings of delays in hierarchical designs.

Sub-Circuit Device Delay

Sub-circuit devices inherit their delay characteristics from their internal circuit and have no "device delay" characteristic of their own. Simulation Parameters cannot be directly used on a sub-circuit device, although pin delays <u>can</u> be set separately on each instance of a sub-circuit device to customize path delays.

Pin Delays

Any input or output pin on any device (including port connectors and sub-circuit devices) can have a pin delay associated with it. Pin delays normally default to 0 time units, but can be in the range 0 to 32767.

A pin delay acts like a "buffer" device with the given delay inserted in line with the pin. On an input pin, the device simulation model will not see a change in signal value until after the pin delay has elapsed. On an output pin, the pin delay is added to the overall device delay for any changes scheduled on that pin.

Effect of zero delay

A delay value of zero is permitted in a DesignWorks device, but this setting should be used only with an understanding of how the simulation is implemented as it can result in unexpected side-effects.

On a given pass through the simulation routine, all the events on the list which occur at the current time are scanned and then the new outputs for all affected devices are calculated. If any of these devices has a zero delay setting then this will result in more changes being placed on the event list at the current time. However, all these changes emerging from zero-delay devices will not be evaluated until the next pass through the simulator. This is done to allow for user interaction with the simulation.

Stepping interactively through a circuit with zero-delay elements results in all these value changes to be updated on the screen, even though "simulation time does not advance. If a signal changes value then reverts to its original state within the same time step, this will be displayed as a zero-width spike in the timing window.

If a zero-delay feedback loop exists in a circuit, the signal changes will be simulated and any probes on the schematic will be updated at each pass through the simulator. However the events at the head of the list will always have the same time value associated with them and the simulated time will never advance. Updating of the timing window will stop until some delay is inserted into the loop.

Where Delays are Stored

Delays are stored as decimal integers in text form in attribute fields associated with each device or pin. For devices, the delay field is called "Delay.Dev", for pins it is "Delay.Pin" An empty or invalid string will be interpreted as the default value, usually 1 for devices and 0 for pins.

Some special-purpose devices, such as the Clock, One Shot and SetupHold primitive devices take two delay characteristics. In this case, two integers separated by a comma should appear in the Delay.Dev field.

NOTE: Note: The Simulation Parameters dialog should be used to control these attribute fields rather than the Schematic tools attribute dialog.

See each of these primitive types in Chapter 11—Primitive Devices on page 129 for more information.

A number of additional fields are pre-defined in DesignWorks for storage of alternate delay values. These fields are not used directly by the simulator, but are only provided as a convenient place to store these values. The Save to Attribute and Set from Attribute functions in the Simulation Parameters dialog can be used to copy values to or from alternate delay fields.

The following alternate fields are pre-defined.

Field Name	Function
Delay.Dev.Typ Delay.Pin.Typ	Typical device/pin delay
Delay.Dev.Min Delay.Pin.Min	Minimum device/pin delay

Delay.Dev.Max Delay.Pin.Max	Maximum device/pin delay

Device Storage State

In DesignWorks, primitive storage devices (e.g. flip-flops, counters and registers) do not store their current state internally. The device state is completely determined by the values on the signals attached to the output pins. Thus, the following factors will affect the operation of these devices:

> Conflicting or overriding values on the output signals (e.g. a stuck state) will override the last device state calculated by the model.
>
> Device and pin delays will influence the calculation of a new device state. E.g. if the period of a clock applied to a counter is less than the total delay through it, an erroneous count sequence will result.

If desired, this behavior can be modified by placing the primitive devices in a sub-circuit device and setting appropriate pin types and delays on the parent device to "buffer" the outputs.

NOTE: These comments do not apply to RAM or bidirectional switch primitives, both of which store internal state information independent of the values of the attached signals.

See "Working With Hierarchical Blocks" on page 82 for more information.

Input Signal Values

Unless an alternate input value mapping is specified (see below), for all device types except switches, signal values "High-impedance" and "Conflict" are treated as "Don't Know" when applied to a device input. When a device is first created, all input signals take the "High Impedance" state and outputs are set depending on their type, normally to the "Don't Know" state. Thus, an unused input pin will appear as an unknown input to a device, which may affect its output level.

As with real circuits, all unused inputs should be connected to a high or low level as appropriate. This can be done by naming the pin signal either "0" or "1", using a power or ground symbol, or by using a pullup resistor to set a high level.

NOTE: Note: “0” and “1” only work for simulation and are NOT equivalent to power and ground for netlist purposes.

Input Value Mapping

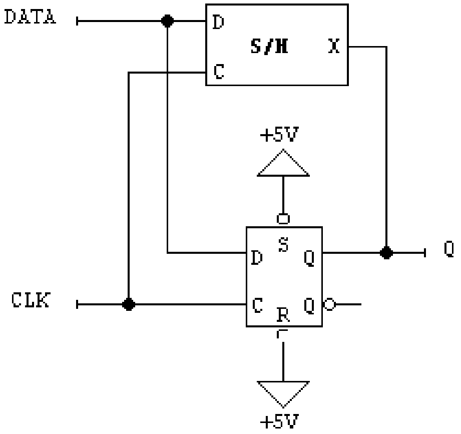
An alternate input value can be specified for a design by placing a specially-formatted string in the Sim.InputMap attribute field for the design. This is intended primarily to model specific logic families that have a known response to high-impedance inputs. The mapping applies only to primitive devices and is global to the design that it is specified in. Note that this mapping does not occur at inputs to sub-circuit devices but does affect the primitives that they contain.

See Appendix C—Simulation Attribute Fields on page 171 for more information on the format of the Sim.InputMap field.

Setup and Hold Times

All standard device types having an edge-triggered clock, such as the D and JK-flip-flops, register and shift register, have an effective setup time of 1 unit and a hold time of zero units. That is, if the data and clock inputs change simultaneously, the old value of the data input will be used.

Setup and hold times can be checked by attaching a "SetupHold" primitive device to the inputs of the clocked device to be checked, as follows:



The SetupHold device puts out a highZ value until a setup or hold violation occurs when it switches to a “Don’t Know” state. Thus, the output of the SetupHold device can be paralleled with the flip-flop so that the output line will enter a conflict state when an error occurs.

See “SetupHold” on page 147 more information on the SetupHold device.

Device Pin Types

Every device pin has a characteristic known as its *pin type*, e.g. input or output. The pin type is set when the part entry in the library is created and cannot be changed for individual device pins on the schematic. Correct pin type settings are crucial to correct and efficient operation of the simulator.

The pin type is used by the simulator to determine the direction of signal flow and which output values are allowable on a given output pin.

See Appendix A—Device Pin Types on page 163 for more information on the available pin types and how they affect the simulation.

See Chapter X - Device Symbol Editing in the DesignWorks/Schematic Reference Manual for procedures on setting pin types when creating a symbol.

Device Pin Inversion

The logic of any pin on any device can be inverted by placing a non-empty value in the Invert.Pin attribute field of the pin. When this is done any value passing into or out from that pin will be inverted. This applies to primitive types as well as sub-circuit devices. The following table summarizes the level mappings that occur.

External Signal Value	**Internal Signal Value**
LOW.H	HIGH.H
LOW.L	HIGH.L
HIGH.H	LOW.H
HIGH.L	LOW.L
All others	Unchanged

NOTE:

1) The logical inversion of the pin is completely independent of the graphical representation of the pin. I.e. using the "inverted pin" graphic in the DevEditor tool does not invert the pin logic in the simulator. The Invert.Pin field must be set to have this effect.

2) Although pin inversion can be specified independently for each device on the schematic we do not recommend modifying these settings after a device has been placed on the schematic. This can create the confusing situation of two devices with the same name and symbol but different logical characteristics.

See also:

"Pin Delays and Inversion" on page 84 for information on pin inversion in sub-circuit blocks.

Chapter 11—Primitive Devices on page 129 for information on how pin inversion can be used with specific primitive types.

"Pin Inversion" on page 175 for more information on pin usage and pin inversion.

Chapter X - Device Symbol Editing in the DesignWorks/Schematic Reference Manual for procedures on setting pin attributes when creating a symbol.

Triggers

The DesignWorks Simulator has a powerful trigger capability analogous to a "word recognizer" on a logic analyzer. Any number of triggers can be set to perform various actions when certain time and signal value conditions are met. These actions include:

Drawing reference lines in the timing window.

Stopping the simulator.

Enabling or disabling the timing display.

Generating a beep from the computers speaker.

Enable checking for another trigger.

Any number of triggers can be set up with different activation conditions. For display purposes, triggers are named T1, T2, T3, etc. To assist in setting up complex time-related conditions, trigger T_N can be used to enable T_{N+1}.

See “Triggers” on page 70 for more information on setting triggers.

Simulation Clearing and Initialization

The DesignWorks Simulator provides a number of mechanisms to assist in setting initial values and restarting a simulation.

The Clear Simulation Operation

The Clear Simulation operation is invoked by clicking on the Reset control in the Simulator window. This operation performs the following steps:

Other tools (such as Timing and Test Vector) are notified and perform their own processing.

All signal change events on the queue are disposed of, whether pending or historical.

Any clocks in the design are re-initialized.

If any signal or pin initial values are specified, they are set up. See below for information on setting initial values.

All devices are queued for immediate reevaluation.

The Clear Unknowns Operation

The Clear Unknowns operation is invoked by clicking on the Clear X control in the Simulator window. This operation performs the following steps, stopping as soon as all unknown states are removed from the design:

Any pending signal change that would result in an unknown state is removed from the queue.

Any primitive type with storage capability (i.e. flip-flop, register or counter) that has a “Don't Know” output value is cleared, either to its specified initial value (if any) or to zero.

The input mapping for the design is temporarily set so that all unknown input values map to zero.

A single device that currently has an unknown output state is randomly selected and queued for reevaluation. The simulator is cycled repeatedly as long as the number of unknown states in the design decreases. This step is then repeated until the number of unknowns ceases to diminish.

The input mapping for the design is restored to its original state.

If this operation does not clear the design to an appropriate state, refer to the other techniques discussed below.

NOTE: Designs with "hard" unknowns, such as unconnected inputs or conflicting outputs will **not** be successfully cleared by this procedure. All device inputs should be specified to a known value if not driven by other devices.

Setting Initial Values

Initial values can be specified for signals and pins which will be applied by the Clear Simulation and Clear Unknowns operations, as described in the preceding sections.

For both object types, the initial value is entered into an attribute field, either Initial.Sig or Initial.Pin. The allowable values consist of a single character chosen from the following table.

Character	**Value**
0	LOW
1	HIGH
Z	HIGHZ
X	DONT01

All other values will be ignored.

NOTE:

1) It is left completely to the user to decide if the specified initial values make sense. E.g. No checking is done to determine if a given device output value is the reasonable result of the device's current inputs.

2) Devices do not have initial value settings since their values are completely determined by the state of their output pins. See “Device Storage State” on page 59

Signal Initial Values

The initial value for a signal is stored in the Initial.Sig attribute field using the format described in the previous section. When a Clear Simulation operation is invoked, the initial value specified is placed on the signal without regard for the current output levels of devices driving the signal. The given value will stay on the signal until some device driving the signal changes state or some other user action changes it.

NOTE: If a pin initial value is specified for any output pin driving the signal, the signal value will be overridden.

Pin Initial Values

The initial value for a pin is stored in the Initial.Pin attribute field using the format described earlier. Initial values can only be specified for output or bidirectional pins and will be ignored on input pins.

When a Clear Simulation operation is invoked, the initial value specified is placed on the pin without regard for the current inputs affecting the device. The given value will stay on the pin until the device model schedules a state change or some other user action changes it.

Using the Simulator

6

Introduction

The chapter describes control of the simulator via the Simulator tool.

Starting the Simulator

By default, the simulator starts running as soon as a circuit is loaded. To display the simulator control window, select “Simulator” from the Tools menu in the DesignWorks window

Simulator Window

The simulator window is a floating window. It is displayed when the Simulator tool is selected from the “Tool” menu in the DesignWorks window OR when a new timing window is created. It can be hidden by selecting “Close” from the system menu in the top left corner of the window.

Time Display

Item	Comment

Time Display	The status area in the top left corner of the simulator window displays the current simulation time as the simulator progresses or “Idle” if the simulator is not running.
Reset	Removes all scheduled signal events, sets all devices, signals and pins to their specified initial values (if any) and recalculates output values for all circuit elements. In addition, it clears the timing window (if open) and resets the Test Vector tool (if open).
Run	Causes the simulator to execute at the fastest possible speed.
Step	Causes the simulator to execute one time step. If the simulator is running then this control causes the simulator to stop after the next event time.
Clear X	Clears all flip-flop, counter and register primitives to the zero state and attempts to remove all unknown signal values from the circuit. Certain circuit conditions may prevent signals from being placed in a known state such as unconnected inputs that have not been set to a known level, storage devices, such as RAMs that have an unknown stored value and/or any simulation models that do not produce a known output when all inputs are known.
Triggers...	Displays the trigger control dialog. See / (page 70) for more information
< (Zoom In)	Increases the horizontal display resolution in the Timing window (if it is displayed).
> (Zoom Out)	Decreases the horizontal display resolution in the Timing window (if it is displayed).
= (Clear Zoom)	Sets the horizontal display resolution in the Timing window (if it is displayed) to its initial defaults.

Simulator Window Menu Commands

Speed Menu

The Speed menu is used to control the simulation speed, i.e. the amount of delay inserted between simulation steps. Simulation speed can be set individually for each open design.

Stop

This command stops the simulator immediately. No simulation processing is done when the simulator is in this state.

Run

This command starts the simulator at the fastest possible speed.

Single Step

This command simulates one time step. To perform the single step, the simulator looks at the time value associated with the next signal change event in the queue, simulates the effect of that and all following events scheduled at the same time, then returns to the stopped state. The actual time value of a single step depends on the nature of the circuit.

Other Simulation Speeds

The intermediate speed settings between Stop and Run insert various amounts of delay between executing successive simulation time steps. These can be used to slow the simulation progress for convenient observation.

Options Menu

Parameters...

This command displays the Simulation Parameters dialog and is enabled when at least one device or pin is selected in the schematic.

See " / " (page 73) in for more information.

Simulation Scope...

This command displays the Simulation Scope dialog.

See " / " (page 78) in for more information.

Stick Signals...

This command displays the Stick Signals dialog.

See " / " (page 79) in for more information.

Simulator Procedures

Triggers

Selecting the triggers control in the Simulator window displays the following dialog:

Trigger Conditions

A trigger is activated when all three sets of conditions are met:

The time condition, i.e. the current simulator time value is less than, equal to, greater than, or a multiple of, a given value.

Signal value condition, i.e. one or more signals are at specified levels.

The delay condition, i.e. the trigger is activated after a certain delay, or the signal condition exists for greater or less than a specified amount of time.

Trigger Enabling

When the "Enabled" switch is on, the trigger is "armed" and the selected actions will take place as soon as the trigger's conditions are met. If this switch is off, this trigger is disabled until enabled by the previous trigger or by this switch being selected.

Time Test Controls

The controls related to the time condition are summarized in the following table.

Control	Comment
Time Value	Enter the time value as a decimal integer. The meaning of this value is determined by the switches below it.
N/A	This control specifies that the time condition should be considered to be always true. The time value is ignored.
Every	This control specifies that the trigger will be activated every time the simulator time equals a multiple of the specified value.
<, =, >	These controls indicate that the trigger will be activated when the simulation time is less than, equal to, or greater than the given value, respectively.

Signal Test Controls

The controls related to the signal condition are summarized in the following table.

Control	Comment
Names	Enter the names of one or more signals whose values will be compared to the hexadecimal integer value typed in the Value field. One or more signals can be entered using the following formats: CLK The single signal CLK D7..0 The signals D7 (MSB), D6, D5 ... D0 IN1 OUT3The signals IN1 and OUT3
Signals Value	Enter the signal comparison value as a hexadecimal integer. This value is converted to binary and compared bit-for-bit with the signals named in the Names field. The rightmost signal name is compared with the least significant bit of the value, etc.

Delay Condition Controls

The controls related to the time condition are summarized in the following table.

Control	Comment
N/A	This control specifies that the delay condition should be considered to be always true. The delay value is ignored.
After	With this control enabled, the trigger will be activated the specified amount of time after the time and signal value conditions are met, regardless of whether they continue to be true.
Sig Stable >	With this control enabled, the trigger will be activated after the specified delay, as long as the time and signal conditions are still true.
Sig Stable <	With this control enabled, the trigger will be activated if the time and signal conditions are true for less than the specified delay. I.e. Activation occurs when the time and signal conditions cease to be true if they have been true for less than the specified value.
Delay Value	Enter the delay value as a decimal integer.

Trigger Actions Controls

When a trigger is activated, any combination of the displayed actions can be invoked.

Control	Comment
Beep	Generates a single system beep.
Stop	Stops the simulator immediately.
Enable T_{N+1}	Enables the next numbered trigger.
Timing Display On	Turns on the timing waveform display.
Timing Display Off	Turns off the timing waveform display.
Reference Line	Draws a vertical reference line at this time on the timing waveform display.

Navigation Controls

The navigation controls perform the following functions:

Control	Comment
OK	Accept changes for this trigger and close the dialog.
Cancel	Discard changes for this trigger and close the dialog.
Delete	Delete this trigger and close the dialog. All subsequent triggers, by number, are decremented.
Prev	Accept changes for this trigger and display the previous trigger (if one exists).
Next	Accept changes for this trigger and display the next trigger.

Simulation Parameters

The Simulation Parameters is a general method of setting device and pin delays and options. If no devices or pins are selected in a circuit then Simulation Parameters menu command will be disabled. This command displays the Simulation Parameters dialog which depends on the types of devices currently selected:

Selection	Dialog	Notes
A single CLOCK device	Clock Parameters	Only one clock device can be set at a time.
A single SETUPHOLD device	SetupHold	Only one SetupHold device can be set at a time.
A single ONESHOT device	One Shot	Only one One Shot device can be set at a time.
Any other combination of one or more devices or pins	General Delay	Any selected CLOCK, ONESHOT, SETUPHOLD, sub-circuit or other non-delay devices will be ignored for device delay calculations.

NOTE: 1) The device delay of a sub-circuit device cannot be set as its general delay characteristics are determined by its internal circuit. If any sub-circuit devices are selected, they will be ignored for device delay purposes. Pin delays on sub-circuit devices can be set to modify the path delay through a particular pin. Delays on the devices or pins inside a sub-circuit device are not affected by any settings on the parent device using this command.

NOTE: 2) The Simulation Parameters rely on numeric information in a specific format being present in the Delay.Dev or Delay.Pin attribute fields. Any invalid information in these fields will be ignored and default values used.

See (page 135) for more information on the usage of specific attribute fields for simulation parameters.

General Delay Dialog

When selection contains devices and pins with delay characteristics, the following dialog is displayed:

Item	Comment
Devices	When this control is enabled, the other controls display and set the *device delay* characteristic of the devices currently selected in the circuit. Items such as CLOCK or sub-circuit devices which have no device delay characteristic will be skipped.
Pins	When this control is enabled, the other controls display and set the *pin delay* characteristic of the pins currently selected in the circuit.
Number of selected devices/pins	Shows the number of devices or pins that will be affected by changes made in this dialog.
Shortest / Longest delay	Shows the shortest and longest delays found in any of the selected devices or pins. (Note that each device or pin has only a single integer delay value associated with it.)
Delay Text	If all selected devices or pins have the same delay value, it is shown here. If a variety of values exist among the selected items, this item will be empty. Typing a new value (between 0 and 32767) will set all items to the given value.

+	This control will add 1 to the delays in all selected items, to a maximum value of 32767.
-	This control will subtract 1 from the delays in all selected items, to a minimum value of zero.
1	This control will set the delay in all selected items to 1.
0	This control will set the delay in all selected items to 0. .
Set From Attribute	This control will display the Set From Attribute dialog, allowing the delays in all selected objects to be loaded from a selected attribute field in each object. This dialog is described below.
Save To Attribute	This control display the Save To Attribute dialog, allowing the delays in all selected objects to be saved to a selected attribute field in each object. This dialog is described below.

See (page 13) for more information on the meaning and usage of device and pin delays.

Clock Parameters Dialog

When a single clock device is selected, the following dialog is displayed:

Item	**Comment**
Low	Enter the low time setting of the selected clock device. Allowable settings are in the range 1 to 32767.
High	Enter the high time setting of the selected clock device. Allowable settings are in the range 1 to 32767.
Set From Attribute	This control will display the Set From Attribute dialog, allowing the parameters in the selected clock device to be loaded from a selected attribute field. This dialog is described below.

Save To Attribute	This control will display the Save To Attribute dialog, allowing the clock parameter in the selected device to be saved to a selected attribute field. This dialog is described below.

See (page 109) for more information on how to set the startup delay and initial value of a CLOCK device by setting the pin delay and inversion on the output pin.

SetupHold Parameters Dialog

When a single SETUPHOLD device is selected, the following dialog is displayed:

Item	Comment
Setup	Enter the setup time setting of the selected device. Allowable settings are in the range 0 to 32767.
Hold	Enter the hold time setting of the selected clock device. Allowable settings are in the range 0 to 32767.
Set From Attribute	This control displays the Set From Attribute dialog, allowing the parameters in the selected device to be loaded from a selected attribute field. This dialog is described below.
Save To Attribute	This control displays the Save To Attribute dialog, allowing the setup and hold parameters in the selected device to be saved to a selected attribute field. This dialog is described below.

See (page 109) for more information on how to set the initial value *of a SETUPHOLD device by setting the inversion on the output pin.*

One Shot Parameters Dialog

When a single ONESHOT device is selected, the following dialog is displayed:

Item	Comment
Delay	Enter the delay time setting of the selected device. Allowable settings are in the range 0 to 32767.
Width	Enter the width time setting of the selected clock device. Allowable settings are in the range 1 to 32767.
Set From Attribute	This control displays the Set From Attribute dialog, allowing the parameters in the selected device to be loaded from a selected attribute field. This dialog is described below.
Save To Attribute	This control displays the Save To Attribute dialog, allowing the parameters in the selected device to be saved to a selected attribute field. This dialog is described below.

See (page 109) for more information on how to set the initial value of a ONESHOT device by setting the inversion on the output pin.

Loading Delays from Attributes

Each of the simulation parameters dialogs described above has Set from Attribute and Save to Attribute controls. Selecting either of these controls displays a dialog similar to the one below:

Set From Attribute

Selecting an attribute field in the list and clicking on the Load control will cause the contents of the selected field to be copied to the Delay.Dev or Delay.Pin field of all selected devices or pins. The copied value will completely replace the old contents. If the selected field is empty in a given object, the Delay.Dev or Delay.Pin field is left unchanged in that object.

Save to Attribute

Selecting an attribute field in the list and clicking on the Save control will cause the contents of the Delay.Dev or Delay.Pin field to be copied to the selected field in all selected devices or pins. The copied value will completely replace the old contents, even if the source value is empty.

Simulation Scope

Selecting the simulation scope command from the Simulator window menu displays the following dialog:

Action	Comment
Enable Entire Design	Removes all previous disables and enables simulation of all parts of the design.
Disable All Levels Above Current Circuit	The current circuit (i.e. the one displayed in the frontmost circuit window) is set to be the topmost level of the hierarchy that will be simulated. Simulation is disabled in all other hierarchy levels and all driving port connectors at this level will drive at a high impedance level.
Disable Internal Circuit of Selected Device	The internal circuit of the device selected on the schematic will be disabled. All the output pins on the device will be set to a high impedance drive level.
Enable Internal Circuit of Selected Device	The internal circuit of the device selected on the schematic will be re-enabled. Its outputs will revert to the state determined by the internal circuit.

Stick Signals

Selecting the stick signals command in the Simulator window menu displays the following dialog:

Control	Comment
Selected Signals in Current Circuit	If this control is enabled, the Stick High, Stick Low or Unstick buttons will apply only to signals currently selected in the schematic.
All Signals in Current Circuit	If this control is enabled, the Stick High, Stick Low or Unstick buttons will apply to all signals in the circuit represented in the topmost schematic window, including circuit pages in the same circuit. If this is part of a hierarchical design, only this circuit level is affected.
All Signals in Design	If this control is enabled, the Stick High, Stick Low or Unstick buttons will apply to all signals in all parts of the current design.
Number of signals selected	The number of signals that will be affected by any changes made in this dialog.
Number stuck high	The number of signals in the selected scope that are currently stuck at a high level.
Number stuck low	The number of signals in the selected scope that are currently stuck at the low level.
Stick Low	This control closes the dialog and applies a "stuck low" value to all selected signals.
Stick High	This control closes the dialog and applies a "stuck high" value to all selected signals.
Unstick	This control closes the dialog and unsticks all selected signals, allowing them to return to their driven value.

See (page 13) for more information on stuck signals.

Using the Simulator with a Schematic Diagram

7

This chapter describes schematic related issues including signal naming, power and ground, simulation models, probing as well as changes to the Schematic tool to support simulation

Schematic Simulation Issues

Hierarchy Mode

The DesignWorks Schematic tool supports three hierarchy modes: Flat, Physical and Pure. The following table summarizes these modes in terms of their significance for simulation.

Mode	Comment
Flat	Despite its name, Flat mode does allow devices to have internal circuits for simulation purposes, although it will discourage access to them. Flat mode also changes the way device names are assigned and reports are generated, but these issues do not affect the simulator.
Physical	This mode allows unrestricted access to internal circuits for simulation purposes.
Pure	Simulation is not supported in Pure mode as only a single definition copy of device type data is kept in memory, regardless of the number of times the device is used. Therefore, there is nowhere to store the simulation states associated with each instance.

Working With Hierarchical Blocks

The simulator does not impose any new rules on working with hierarchical blocks, but there are some effects of editing a design with active simulation that should be noted.

See Chapter VII - Hierarchical Design in the DesignWorks/Schematic Reference Manual for additional information.

Editing an Open Internal Circuit

A number of issues arise if the same device type or hierarchical block appears multiple times in a design and one copy of the type or block is opened for editing (i.e. using the Push Into command or by double-clicking on the device):

> The Schematic tool creates a separate, temporary type definition for the open device when it is opened. Any simulation values viewed or changed, or any circuit changes made, will apply only to that one device instance while it remains open.
>
> When the open internal circuit is closed, the action taken depends on edits that have taken place. If no edits have been made or changes have been made only to instance data (i.e. simulation values or attributes marked as "keep with instance"), then the other blocks of the same type will not be affected by the close. If any definition data (i.e. any graphical or structural change to the circuit or any definition attribute change) has changed then a dialog is displayed prompting for the action to take. If the Update control is selected then all instance data (i.e. signal values, etc.) in other blocks of the same type will be lost. Instance data will be completely replaced by the values from the edited block.

The Port Interface

The connection between a pin on a parent device symbol and the corresponding signal in the internal circuit is quite complex from a simulation standpoint. In order for this connection to act like a "hard wire" between the two levels, the following conditions must be met:

> the *pin type* on the parent device symbol must be "bidirectional".

the *pin type* of the corresponding port connector in the internal circuit must be "bidirectional".

the *pin delays* on both the pin on the parent device and the pin on the port connector must be zero.

no *pin inversion* must be specified either on the parent device pin or the port connector pin.

Any other combination of settings will result in some degree of isolation or "buffering" between the two levels. I.e. The observed signal value on the signal in the internal circuit may be different from that on the parent pin.

NOTE: NOTE: When a symbol is created in the DevEditor tool, all pins default to type "input", i.e. they will not drive any attached signal. When creating a hierarchical block symbol for simulation purposes, the pin types must be set to appropriate values.

The effects on these various settings are summarized in the following sections.

Parent Device Pin Type

Any signal value driven out of a parent pin by an internal circuit may be translated according to the pin type on the parent device. These effects are summarized in the following table.

Pin Type	**Effect**
Input	This will prevent that pin from ever driving the attached signal, regardless of drives in the internal circuit.
Output / Three-state	This will pass the sum of the internal drives up to the parent pin without any translation. Signal value changes on the signal attached to the parent pin will not be passed to the internal circuit.
Open collector / Open emitter	Any drive level from the internal circuit will be translated according the capability of the pin type.
Bidirectional	All changes on the internal signal are passed to the parent pin and vice versa.
Other types	Other types, such as Tied High and Tied Low are not recommended.

NOTE: NOTE: Although it may be tempting to set all pins to "bidirectional", this is not recommended since it significantly increases simulation overhead and increases the difficulty of isolating circuit drive problems.

Port Connector Pin Type

The pin type on the port connector is also used to translate the value of any incoming signal changes, in a manner similar to the parent pin type. Normally, the pin type setting on a port connector should complement the setting of the parent pin as follows:

Parent Pin Type	Port Connector Name	Port Connector Pin Type
Input	Port In	Output
Bidirectional	Port Bidir	Bidirectional
All others	Port Out	Input

Other settings on the port connector pin are not recommended.

Pin Delays and Inversion

The normal pin delay and inversion settings can be applied to the port interface. A non-empty value in the Invert.Pin attribute field will cause any signal values passing in either direction to be inverted. An integer value in the Delay.Pin attribute will cause the specified delay to be inserted in line with level changes passing in either direction.

NOTE: 1) It is recommended that pin delay and inversion settings be applied only to the pin on the parent device and not to the port connector in the internal circuit. Attribute settings on the port connector are more difficult to verify and edit since the port connector is a "pseudo-device" and some schematic editing operations will be disabled.

2) Changes made in Invert.Pin and Delay.Pin attribute field after a device has been placed on the schematic, will affect only that one device instance. Default values can be set in these attribute fields when the symbol is created in DevEditor tool.

Power and Ground Connectors

Power and Ground connector symbols do not have any inherent simulation signal drive unless their pin type has been set to "Tied High" or "Tied Low", as appropriate. Most of the positive-supply symbols provided with this DesignWorks have "tied high" settings, while others will be "tied low". The symbols provided with older DesignWorks releases may not have any drive setting, resulting in a high impedance level on these signals. This can be remedied by either:

Replacing any one or all of the ground or power symbols with one that has the appropriate setting.

Forcing a Stuck High or Stuck Low level onto the signal using the probe tool, Test Vector tool or other external means. Note that because all like-named ground or power segments are logically connected, this only needs to be done on any one segment.

Special Signal Names 0 and 1

The signal names "0" and "1" are recognized by the simulator as special. If any signal is named "0", it will be given a Stuck Low value. If "1" is found, it will be given a Stuck High value. These values can be cleared or changed using the probe tool, if desired.

See “Probing a Signal” on page 87 for more information.

Simulation Models

In order to completely simulate a design, every symbol must have an associated simulation model. In DesignWorks, simulation models can take one of the following forms:

Primitive Devices - These types have "hard-wired" program code to evaluate input and output changes. They include the gates, flip-flops and other devices as described in , as well as the user-definable PROM and PLA primitives.

Sub-circuit Devices - The simulation function of a sub-circuit device is completely determined by its internal circuit (except for the addition of pin delays and inversion). The definition of a device sub-circuit can be stored with the part in a library (referred to as an "internal sub-circuit") or can be stored in a separate design file on disk (an "external sub-circuit"). The sub-circuit itself can contain any combination of primitive devices or other sub-circuits (except itself, of course!) nested to any desired depth.

Whenever any device type is to be simulated, all information about the device must be loaded into memory. Unless internal circuits or code models are explicitly purged from the design, they will become a permanent part of the design and will be saved with the file.

Primitive Devices on the Schematic

Primitive devices are provided in various primitive libraries and can be used at any time as part of a schematic, whether or not the simulator is installed. However, these libraries are not intended to match any real logic families and do not have any part name, pin number or packaging information associated with them.

See Chapter 11—Primitive Devices on page 129 for more information on creating and using primitive types.

Simulation Pseudo-Devices

The simulation pseudo-devices (i.e. those in the Primitive I/O library) are handled specially by the Schematic tool. In general, the symbol, pin types or other characteristics of these devices cannot be modified. In addition, they are treated differently from normal device symbols in the following ways:

By default, these devices are flagged "omit from report", meaning that they will not appear in any netlist or bill of materials produced by the Report tool. This setting can be changed using the Schematic tool's Get Info command.

These symbols will not be assigned a name when placed on a schematic. A name can be manually assigned, if desired.

The switch and keyboard types respond to a normal mouse click by changing state, rather than being selected. To select one of these devices hold the shift key down while clicking on it.

Using External Sub-Circuit Models

Most of the models for industry-standard device types (e.g. 74XX, 4000, etc.) are provided as external sub-circuits. These models are simply design files containing an equivalent circuit for the device. The circuits are designed to match with the symbol libraries provided with the Schematic package.

External sub-circuit models can be loaded either interactively, i.e. as each part is used, or in batch mode after the design is created. Both of these functions are provided by the SimLoad tool.

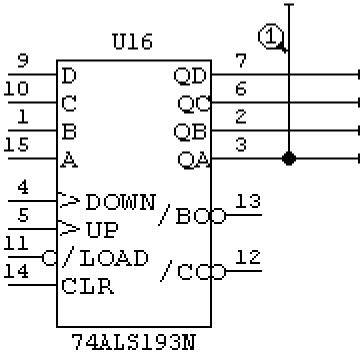
See "Automatic Model Loading" on page 89 for more information.

Probing Signals / Schematic Tool Palette

When the Simulator Tool is installed, a probe tool in available in the palette window displayed by the Schematic tool. Selecting the Probe tool places the schematic in Probe mode, allowing signal and pin values to be displayed and injected. When the probe tip is clicked and held on a signal or pin, the cursor will show the current value on the signal or pin and track changes that occur as the simulation progresses.

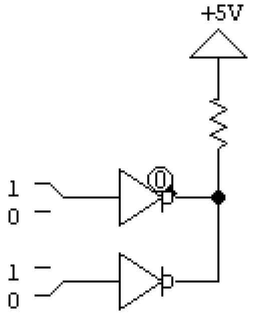
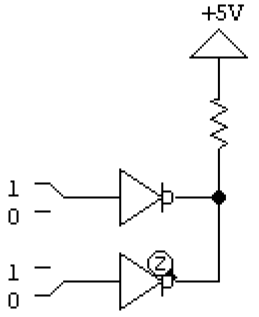
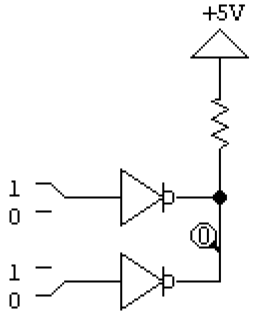
Probing a Signal

Only the signal under the cursor at the time of the click is examined. Moving the mouse while the button is pressed does not change the signal being viewed.



Probing a Pin

If the probe tip is clicked on a device pin close to the device body, the probe shows the driving level of that pin, rather than the state of the attached signal. This can be used to resolve drive conflicts in multiple drive situations, as in the following example:

Pin Drive on Upper Device	Pin Drive on Lower Device	Combined Signal Value
		

NOTE: The probe display does not distinguish between low and high drive levels.

Injecting a Value Using the Probe Tool

While the mouse button is held, press the appropriate key to inject a new value onto a signal:

Key	Injected Value
0	LOW.F
1	HIGH.F
X	DONT01.F
C	CONF.F
Z	HIGHZ
L	LOW.F stuck
H	HIGH.F stuck
space	Unstick

If a stuck value is forced onto a signal, it will not change state until the stuck value is cleared by some user action, regardless of device outputs driving the line. If a non-stuck value is forced, the signal value will revert to its appropriate new level when any change occurs on a device output driving the line.

The spacebar "unstick" command causes the signal to revert to its driven value.

NOTE: Note: values cannot be injected onto pins.

See “Stuck-at Levels” on page 54 for more information.

Automatic Model Loading

This chapter describes automatic or manual loading of simulation models using the SimLoad tool.

Model Loading Options

The SimLoad window is a dialog. It is displayed when the SimLoad tool is selected from the “Tool” menu in the DesignWorks window. It can be hidden by selecting Cancel or Done from the window controls.

SimLoad Window Controls and Fields

Status

Most of the models for industry-standard device types (e.g. 74XX, 4000, etc.) are provided as external sub-circuits. These models are simply design files containing an equivalent circuit for the device. The circuits are designed to match with the symbol libraries provided with the Schematic package.

External sub-circuit models can be loaded either interactively, i.e. as each part is used, or in batch mode after the design is created. SimLoad defaults to the inactive state, i.e. no loading occurs.

Two levels of loading action are available:

> References Only - This method can be used to check for the existence of the model file without actually loading it. In this case, each time a new part is used, SimLoad attempts to locate a model for it and sets the external circuit reference attributes to the location of the file. The file itself is not loaded, i.e. the device will not have a sub-circuit. The sub-circuit can be loaded later using the Attach External Sub-Circuit command in the Sub-Circuit sub-menu in the Schematic tool's Options menu.
>
> Loading On - In this mode, SimLoad locates the external model file and immediately loads and attaches it to the part. The external reference attributes are also set, allowing for later updating if the model file is changed.

NOTE: NOTE: The SimLoad action in both the above cases is the same. In "Loading On" mode, the SimLoad tool directs the Schematic tool to enable its "Auto-Load External Sub-Circuits" mode. This mode can also be controlled directly in the Sub-Circuit sub-menu in the Schematic tool.

Warnings

This section allows the warning level to be set in the event that a model cannot be located during interactive loading.

Control	**Comment**
None	No warning is issued.
Beep	A standard beep tone will be issued each time a new device type is used for which a model could not be located.
Alert	A warning dialog will be displayed whenever a model is not located.

It is possible to determine at any time which devices have an external model associated with them by viewing the contents of the ExtCctName attribute field. This can be done using the Schematics Attributes dialog, the Browser tool (select the ExtCctName field in the Secondary field list), or using the Report tool. A special report format file called "External Circuit Form" is provided with the simulator for this purpose.

Logic Families

This section can be used to restrict the types of devices that will have models attached to them. When the "All Types" control is selected, every device that is used that is a sub-circuit primitive type and does not have an internal sub-circuit will result in a search for a model to attach. When the Select Types control is selected, only the specific standard families indicated will be checked for external models.

Model Directories...

By default, the SimLoad tool searches for simulation models in the directories displayed in this dialog:

To add a new directory, click on the Add Directory control. This will display a standard directory selection dialog. To remove a directory from the search list, simply select it in the list and click on the Remove Directory control. The default directory can be added by clicking on the Add Default control.

NOTE: Note: The default directory must be included in the list of directories in order for it to be searched.

Attach Models...

SimLoad allows batch loading of external models. The Attach Models control will display the following dialog:

The Attach Models Scope section allows the scope of the search to be specified. When the Done control is clicked, the devices in the given scope are checked, external references are created or updated, where possible, and the external circuits are loaded and attached. After the operation is complete, a summary dialog will be displayed indicating the number of models found and loaded.

NOTE: This operation can result in a dramatic increase in the amount of memory used by the design. If insufficient memory is available, a message will be displayed.

Detach Models...

SimLoad's Detach Models operation performs the reverse operation to Attach Models. Devices in the selected scope are scanned and the internal circuit is removed from every device having an external reference. The external reference attributes are left intact, allowing for later reloading of the external models.

SimLoad Procedures

Checking Individual External References

The Attach External Sub-Circuit command in the Sub-Circuit sub-menu of the Options menu in the Schematic tool is useful for checking or editing the external reference attributes of a single device. Select the device in question in the schematic, then select this command. It will show the current status of the external reference and allow it to be updated.

Batch Updating or Reloading Models

When an external sub-circuit is loaded, the Schematic tool stores the "Last Modified Date" of the file in the ExtCctDate attribute field of the device. This allows it to check the status of external files relative to a device's internal circuit. Devices that have existing sub-circuits that were loaded from external models, or that have an external reference but no internal circuit, can be updated using the Update External Sub-Circuits command in the Sub-Circuit sub-menu of the Options menu in the Schematic tool. This command will report if all selected items are already up to date.

Creating External Sub-Circuit Models

External sub-circuit models are simply normal DesignWorks design files with port connectors added to match the pins on the parent device. Following is a brief summary of the considerations affecting sub-circuits:

In order to be loaded automatically using the SimLoad tool, the file must be named according to the conventions outlined later in this chapter.

A port connector should be placed in the sub-circuit for each pin on the parent symbol, including no-connect pins. Failure to do this will result in the Schematic tool reporting an error each time the sub-circuit is opened and closed.

Each port connector must be named to match the corresponding pin name on the parent device.

The type of each port connector should match the type of the corresponding pin on the parent device. E.g. An input pin on the parent device should match with a "Port In", or other appropriate type.

The sub-circuit can contain any combination of other devices, including nested sub-circuits to any depth. The only restriction is that the sub-circuit may not contain the parent type or any other type that would create a recursive nesting of blocks.

See “Working With Hierarchical Blocks” on page 82 for more information.

See Chapter VII - Hierarchical Design in the DesignWorks/Schematic Reference Manual for other considerations relating to the creation of sub-circuits.

Customizing Delays for Logic Families

For cases such as the 74XX families, only one model is provided for all logic families (e.g. 74ALS, 74LS, 74HC, etc.).

NOTE: These models provide logical functionality only. No attempt is made to match the delay characteristics of the individual parts.

Delays can be manually customized to match a specific part type by either:

Modifying the sub-circuit model (either in the model file or after it has been used in the design), OR

Setting appropriate pin delays in the parent device.

Model File Naming Conventions

When searching for an external sub-circuit model to attach to a device, the SimLoad tool generates a file name based on the type name, i.e. the name as it appears in the parts palette. The specified search directories are then scanned for the generated file name.

File names are generated differently for each common logic family type, using the following rules:

File names always end with ".EXT" (note that file names are not case-sensitive).

Any text in parentheses is ignored, since these are assumed to be package codes.

If the first two characters of the name are "54" or "74" the a 54XX or 74XX logic family is assumed. Any alphabetic characters are skipped (i.e. the family type) and the file name "74_nnn.EXT" is generated, where "nnn" refers to any remaining digits that are found.

If the first character of the name is "4" a 4000-series part is assumed. Any alphabetic characters are skipped and the file name "4_nnn.EXT" is generated, where "nnn" refers to any remaining digits that are found.

If the first character of the name is "1", then a 10K or 100K ECL part is assumed and the name "10_nnn.EXT" or "100_nnn.EXT" is generated, as appropriate.

For all the above cases, if the name ends with "M", it is assumed to be a "monolith" symbol (i.e. with multiple gates in one symbol) and the M replaces the underscore character. e.g. "74M00.EXT".

For all the above cases, if the name ends with a gate unit extension (e.g. "74ALS240.a"), then the unit letter replaces the underscore. e.g. "74a240.EXT".

For all other cases, the type name (with text in parentheses stripped) is used verbatim (up to 8 characters).

Using VHDL 8

This chapter describes how to use the VHDL language to create design descriptions and device models in DesignWorks.

VHDL Implementation in DesignWorks

A major new feature in DesignWorks is the ability to create simulation models using the VHDL language. However, in order to provide a useful product while still being able to sell it at a reasonable price, we have omitted some of the more advanced features of the language in this version. This note lists the language features that have been omitted.

NOTE: DesignWorks does not implement 100% of the VHDL standard language specification. We have tried to provide a useful set of features that will allow small to medium scale designs to be fully implemented and allow the concepts and exercises in commonly-used textbooks to be executed

The DesignWorks package is being continually upgraded. Please check our web site at www.DesignWorks5.com for a free update patch to the latest version, which may have eliminated some of these restrictions.

VHDL Keywords Not Implemented

The language features associated with the following keywords are not implemented:

```
ACCESS
ALIAS
ATTRIBUTE
```

```
BUFFER
CONFIGURATION
DISCONNECT
FILE
GENERATE
GROUP
LABEL
NEW
PORT MAP on BLOCK
POSTPONED process
PROTECTED
REAL
RECORD
SHARED
TRANSPORT/INERTIAL/REJECT
UNITS
'STABLE 'QUIET 'TRANSACTION and 'DELAYED
```

Other VHDL Features Not Implemented

In addition, some specific language features, or unusual operations on some data types have not been implemented, including:

named associations in aggregates

operator overloading

extended identifiers

physical type declaration

enumerated character type declaration

user-defined resolution functions

the IEEE standard libraries are not 100% implemented

attributes: only the following attributes are implemented:

```
o      RANGE
o      LENGTH
o      LEFT
```

Standard Libraries Supported in DesignWorks

DesignWorks supports only the following standard libraries:

```
SYSTEM
STD_LOGIC_1164
STD_LOGIC_ARITH
```

The above packages are hardwired into the compiler for maximum speed and compactness. In general, other packages cannot be used without some massaging because the standard source code uses VHDL features that are not yet implement in the package.

Other Implementation Notes

Compilation Units

The VHDL compiler in DesignWorks will only accept a single ENTITY/ ARCHITECTURE pair or PACKAGE/PACKAGE BODY pair per file.

Signal Values in VHDL Simulations

When you compile a VHDL file in DesignWorks, the program actually generates a structural circuit consisting of the various blocks described in the code. Each process (including signal assignments or other constructs that create an implicit process) creates a device with signal inputs and outputs. The existing DesignWorks simulator simulates the signal interconnections between these process blocks. This gives considerable flexibility in interconnecting structural and VHDL components, but places some restrictions on signal values. See the section Interface Between VHDL and DesignWorks for more details.

The Timing Diagram 9

Introduction

This chapter describes displaying of timing waveforms via the Timing Tool.

Starting the Timing Tool

To start the Timing tool, select it from the Tools menu in the DesignWorks window. Only one Timing window can be displayed per design, regardless of the number of hierarchy levels in the design.

Timing Window

The timing window is displayed when the Timing tool is selected from the "Tool" menu in the DesignWorks window. It can be hidden by selecting "Close" from the system menu in the top left corner of the window.

Trace data area

Trace name area

Status area

Horizontal sroll bar

Time scale area

Vertical
scroll
bar

Item	Comment
Time Scale	Located just below the timing window's title bar, the time scale is used to establish the absolute timing of value changes in the trace area. The scale is dependent upon the timing resolution (set using the < and > controls in the simulator window). The time scale is also used to set insertion points and selection intervals for use in editing functions.
Trace Data Area	Displays the simulation results.
Trace Name Area	Displays the list of trace names corresponding to the timing traces on the right. Traces can be repositioned by dragging them vertically in this area. In addition, a pop-up menu can be displayed by pressing the right mouse button on a signal name.
Status Area	This area shows progress information for timing commands.
Horizontal Scroll Bar	This control allows scrolling of the trace area. The horizontal scroll bar is disabled when the simulation is running.
Vertical Scroll Bar	This controls allows scrolling of both the trace name and data areas.

Timing Window Menu Commands

File Menu

Import Timing (Text) ...

This command clears the timing window, prompts for a timing text file, opens the file and, finally, pastes the data into the trace area. This is equivalent to selecting the Clear Simulation command, then using the Paste command to place the file data at time zero. See the section “” (page 112) for more information.

NOTE: NOTE: This command does not display or remove any traces in the timing window. It only reads signal event data and associates it with matching traces. If any traces are named in the file that are not currently displayed, you will be warned and that data will be skipped.

See (page 67) for a description of the file format.

Export Timing (Text) ...

This command prompts for a save file then saves all the displayed trace data to a supplied file. This file can be used for external purposes, or can be reloaded as a setup for a new simulation using the Import Timing (Text) command.

See (page 67) for a description of the file format.

Print Timing...

This command displays the print dialog to initiate printing of the timing windows contents using the current printer setup. The current display will be divided into as many pages as required.

Page Setup...

This command displays the page setup dialog which is used for the Print Timing command. This page setup can be different than the schematic diagram page setup.

Edit Menu

Undo

This command undoes the last editing operation in the Timing window. Unlike the Schematic tool, Timing supports only a single Undo and no Redo operation.

Copy

This command copies the selected timing data to the clipboard in picture and text format. The data is stored in both picture form and text data form (i.e. Test Vector command format).

See “ / ” (page 110) in for more information.

See (page 67) for a description of the file format.

Paste

This command pastes the timing text data from the clipboard onto the selected area of the timing window. The selected time interval is deleted and then the new data is inserted. i.e. data following the selection interval will be moved forward by the width of the selection interval, then back by the width of the pasted data.

See “ / ” (page 110) in for more information.

See (page 67) for a description of the file format.

Select All

This command selects all traces and the entire time interval of the timing display.

Find ...

This command allows you to locate a signal or group in the timing display by its trace name. The following dialog will be displayed:

The name of any displayed trace can be typed into the Signal control. When the Find button is clicked, the item will be located and selected (if necessary, the window will be scrolled vertically to display it). The search is done by the displayed trace name in the timing window, not the original name of the item in the schematic.

The following "wildcard" characters can be used in this search:

Wildcard	Comment
*	Matches any string of zero or more characters. For example "D*" will match "D", "D0", "D12", "DogGone", etc. "*" by itself matches anything.
#	Matches any string of 1 or more digits. E.g. "XA#" will match "XA0", "XA123", etc.
?	Matches any single character. For example, "CTRL?" will match "CTRLX", "CTRL/", etc.

The wildcard characters can be used in combination, as in "DATA#??" which will match "DATA123", "DATA1SQ", etc.

Timing Menu

Display On

This command enables updating of the timing display.

Display Off

This command disables updating of the timing display. Events are saved but are not drawn into the timing window. This allows simulation to proceed at a substantially faster rate. Updating of the timing window can be re-enabled either manually or by a trigger.

Normal Size

This command sets the horizontal display resolution to its initial defaults.

Enlarge

This command increases the horizontal display resolution in the timing window.

Reduce

This command decreases the horizontal display resolution in the timing window.

Timing Options ...

This command displays the following dialog:

Timing Data Retention

These options allow you to determine how much signal event data is retained in memory when a simulation is run.

Each time a signal level change occurs DesignWorks creates a record in memory containing a reference to the signal, time, new value and source of the change. In a large simulation these records can consume enormous amounts of memory. This data can be retained for the following purposes:

for use in refreshing the timing window should it become hidden then redisplayed.

for use in timing window editing operations such as taking the output from one circuit and using it as stimulus for another.

Data can be retained only for signals displayed in the Timing window. Signal event data for all other signals is discarded immediately after it is no longer required for simulation.

The option "Retain for Displayed Range Only" is the normal default and results in data being discarded immediately after the corresponding point on the timing display scrolls off the left hand side. This results in minimal memory usage. The setting is equivalent to entering 0 in the retain time box.

The option "Retain for x Time Units" allows you to keep the signal event data for the specified amount of time after it scrolls off the left side of the screen. If this results in a memory shortage occurring then the simulation will stop and a message will be displayed.

Signal Name Display

This option lets you choose how names of signals in sub-circuits will be displayed in the timing window. The option "Show Hierarchical Names" will display the complete "path" to the signal, i.e. prefixed with the names of all parent devices. The option "Show Simple Names" will show only the basic signal name in the trace name area of the window.

Signal Menu

Get Info ...

For groups, this command displays a dialog allowing reordering of the signals in the group. This affects the way the combined hexadecimal value is shown in the timing display.

For individual signals, a signal info dialog is displayed.

Go To Schematic

This command selects the signal or groups in the schematic module corresponding to the first highlighted signal or groups in the timing window then brings the required schematic window forward.

Remove

This command removes the selected signals or groups from the timing window.

Group

This command combines all the selected traces into a single display group. If any of the selected traces were already grouped, they are in effect Ungrouped first and then recombined with other selected items into a single new group.

Ungroup

This command breaks all signals in selected groups into individual traces.

Timing Window Pop-Up Menu

Clicking the right mouse button in the trace name area of the timing window will display the timing pop-up menu.

Get Info ...

For groups, this command displays a dialog allowing reordering of the signals in the group. This affects the way the combined hexadecimal value is shown in the timing display.

For individual signals, a signal info dialog is displayed.

Go To Schematic

This command selects the signal or groups in the schematic module corresponding to the first highlighted signal or groups in the timing window then brings the required schematic window forward.

Remove

This command removes the selected signals or groups from the timing window.

Group

This command combines all the selected traces into a single display group. If any of the selected traces were already grouped, they are in effect Ungrouped first and then recombined with other selected items into a single new group.

Ungroup

This command breaks all signals in selected groups into individual traces.

Collect

This command brings all selected items together in the display underneath the topmost selected item. This is used to bring associated signals closer together for easier comparison of timing, etc.

To Top

This command sends all selected traces to the top of the timing window.

To Bottom

This command sends all selected traces to the bottom of the timing window.

Timing Procedures

Repositioning Signals

Any collection of selected trace names and their corresponding timing data can be repositioned within the list by clicking on the desired trace names in the trace name area (hold the shift key down while clicking to select multiple items) and, keeping the mouse pressed, dragging the vertical line to its new location. Releasing the mouse button will cause the list to be revised with both trace names and trace data traces in their new positions. Alternatively, the To Top, To Bottom and Collect commands in the pop-up menu can be used.

Displaying Groups

The Timing tool allows multiple signal lines to be grouped into a single trace with values displayed in hexadecimal.

Creating a Group Trace

A group trace can be created by any of the following methods:

Select traces by clicking on the desired names (hold the shift key down while clicking to select multiple items). Use the right mouse button on any item then select the Group command in the pop-up menu.

Select any collection of signals in the schematic diagram, then select the Add As Group command from the Timing menu in the Schematic window.

NOTE: NOTE: If any of the selected traces is already displayed, it will not be added to the group.

Select a bus in the schematic diagram, then select either the Add To Timing or the Add As Group command in the Timing menu. Busses are added as a group by default. They can then be ungrouped if desired, using the Ungroup command in the timing pop-up menu.

Order Within a Group

For the purposes of displaying a hexadecimal value for a group, the order of signals within the group is important. When a group is created, the following rules are used to establish the order:

If the signal name has a numeric part (e.g. "D12" or "WRDAT4X", then the numeric part is used to sort the signals. The lowest numbered signal will be the least significant bit of the group value. Any unnumbered signals will be in the most significant bit positions.

Otherwise, the signal's existing position is used, i.e. traces that appeared higher in the timing window will be more significant.

The order of signals within a group can be changed using the Get Info command on a group trace. This is displayed by selecting the Get Info command in the timing pop-up menu or by double-clicking on the trace name.

Entering a Group Name

When a group is first created, a group name is automatically generated from the names of the enclosed signals. This name can be edited using the Get Info command in the timing pop-up menu.

NOTE: NOTE: The group name is lost when an Ungroup operation is done.

Copying and Pasting

To select timing data for copy operations the simulation must be stopped. To do this:

Click in the trace area or timing scale area of the Timing window.

Click on the Step control in the Simulator window.

Select Stop from the Speed menu in the simulator window.

There are two methods of selecting areas for edit operations.

Separate Trace Name and Interval Selection

Select traces to be included by holding the shift key down while clicking in the trace name area, then select the time interval is selected by clicking and dragging in the time scale. This allows you to select non-contiguous traces in the display.

Click on the desired name in the trace name area to highlight and select it. To select more than one name, hold the shift key down and click on the labels.

To set the selection interval, click and hold down the mouse button in the time scale at either end of the desired interval. Drag left or right until the desired interval is enclosed. When the mouse button is released the select interval is set and two selection interval lines will appear. If any of the trace names were selected, the timing signal within the selected interval will be highlighted in the time display.

NOTE: NOTE: Clicking and releasing the mouse button at one spot will create a zero-width interval. This can be used to insert Pasted data without deleting any existing data.

Drag Selection

This method allows you to select a group of trace names and a time interval in a rectangular area of the timing window.

To do a drag selection, click and hold the mouse button at any corner of the rectangular area you wish to select. Drag diagonally across the desired area. When the mouse button is released, the enclosed time interval and traces will be selected.

NOTE: NOTE: The selection operations in the timing window have no effect on selections in the schematic window.

Selecting All Traces or All Time

To select a specific time interval in all traces on the diagram:

Use the Select All command in the Edit menu to select the entire diagram.

Drag select an interval in the time scale area <u>without clicking in the trace data area</u>.

To select all time intervals for specific traces:

Deselect all traces (see below) then hold the shift key down and click on the desired trace names to be selected.

Deselecting

Click in the trace name area above or below the name list to deselect all signals.

Paste Rules

The following rules are used for matching the data on the clipboard with the selected interval in the timing window:

Data is always pasted by name, i.e. the name of a signal in the clipboard data will be matched by the same-named signal in the timing window. Neither the order of the signals in the clipboard data or the selected status of traces in the timing window is significant. To paste data from one signal to a signal with a different name, it is necessary to paste it first into the Test Vector or into a text editor, modify the names, then paste it back.

The Paste operation affects only signals named in the clipboard data, regardless of the selection in the timing window.

The Paste operation will not locate signals in the schematic that are not currently displayed in the timing window. No new traces will be added by this operation.

If the time interval selected in the timing window is non-zero width then the selected interval is deleted and all later events on pasted signals are moved forward. A time interval equal to the width of the clipboard data is then inserted and the new data pasted into this interval.

The Test Vector Tool

10

Introduction

This chapter describes how to use the Test Vector Tool to apply sequences of input values to a circuit and check for the desired result. This tool allows you to verify that circuits are performing as desired without having to manually enter values and check results.

Starting the Test Vector Tool

The Test Vector Tool can be started manually using the available controls and menu items or can be started automatically when a simulation is started.

Starting a Test Run Automatically

The Test Vector Tool starts automatically when you start a simulation if an appropriate test file is located. The test file is a text file containing commands which are described later in this chapter. The file must have the same name as the current design file except with the file extension ".tsv" and must be located in the same directory.

Displaying the Test Vector Panel

The Test Vector Panel displays the exectution and results of a test vector run and has controls to start the execution of a test file. To display the panel, select the Simulation/Test Vectors/Show Test Vector Panel command.

Manually Starting a Test Run

To start a test run, either,

Display the test vector panel as described above, click on the Run button, and select the desired input file, or,

Select the Simulation/Test Vectors/Run Test Vector File command and select the desired input file.

Re-running the Save Test File

To re-run the last file run, click the Run Again button on the test panel, or select the Simulation/Test Vectors/Run Test File Again menu command.

Test Vector Commands

Row and Column Usage

In the spreadsheet area, each horizontal row usually corresponds to one "test vector", i.e. a set of signal stimuli and comparison data that is applied at a particular time step in a simulation. Until the limit of 32,767 rows is reached, you can always enter new data into the empty rows displayed at the bottom of the spreadsheet.

The cells in the spreadsheet have no fixed meaning. Their usage at any given time depends upon the last command that was encountered in that column. Normally, we recommend using separate columns for inputs, outputs, and commands, but it is quite possible to mix them in a single column.

$Commands

Test Vector commands always consist of a "$" character followed by a keyword. Commands fall into two groups:

Definition Commands indicate how the following cells in the same column are to be interpreted. These usually give a list of signal names whose values will be specified or observed. Definition commands have no immediate effect on the execution of the test program or on the circuit under test.

Control Commands allow specific loop control, delay or other control actions to be specified.

NOTE: For compactness, command keywords can always be shortened to the minimum number of letters needed to distinguish them from other commands. In most cases, only one letter is needed, e.g. "$I" can be used instead of "$INPUTS". The minimum contraction is shown in the command list below.

Note the following rules concerning command execution:

Commands have to be "executed" (i.e. stepped through) before they take effect. E.g. if you select the cell under a command and then step downwards from there, the command will have no effect.

Non-command cells (i.e. anything not starting with a "$") will be ignored completely unless a previous definition command indicates how to interpret them.

A definition command in a given column always completely overrides any previous definition command in the same column.

If the circuit schematic is modified while a test program is in progress, all current signal lists are invalidated and no signal data will be interpreted or displayed until the next definition command is encountered.

Definition Commands

NOTE: The $DELAY command can be used either as a control command (if a delay value follows the command keyword) or definition command (if the $DELAY command appears alone in a cell). It appears in both sections below.

Command	**Short Form**	**Description**
$TIME	$T	Display current simulation time in the following cells.
$INPUTS signalList	$I	Specify the values of input signals.
$OUTPUTS signalList	$O	View the values of the given signals (can actually be used to view the "real" values of inputs as well).
$EXPECTED signalList	$E	Specify the expected values of the given signals
$WAIT signalList	$W	Pause the test program until a specified value is detected.
$DELAY	$D	Delay by the specified amount before executing this line.

Signal List Format

Many of the definition commands require that a list of signals be specified. The format of this list can be seen in the following examples:

CLK

specifies simply the single signal "CLK". In this case, each test vector will consist of a single entry, e.g. "0" or "1".

A B C D

specifies four separate signals A, B, C and D. In this case, each test vector will consist of four values separated by spaces, e.g. "0 1 1 0".

[A B C D]

uses the grouping operator [] to indicate that the four signals will be specified by a single hexadecimal number with A in the most significant bit position. E.g. the hex test vector "E" (or 1110 binary) would assign a 1 value to A, B, and C, and a 0 value to D.

NOTE: For compatibility with the Timing tool, Test Vector will allow a "group name" to be prefixed to a group, e.g. "CONTROL[A B C D]". This group name is not used internally and will be ignored. For this reason, there must be a blank between any preceding signal name and the opening "[" or the name will be ignored.

```
[D3..0]
```

uses the ".." operator to indicate a sequence of numbered signals. This case also uses the [] to indicate that the four signals D0, D1, D2 and D3 are to be treated as a single integer, in this case with D3 in the most significant position.

```
CLK CLR [D15..0] [FC2..0]
```

specifies four groups, the first being the signal CLK, the second being the signal CLR, the third being the numbered signals D0 through D15, and the last being the set FC0, FC1 and FC2. A typical test vector for this group might be:

```
.C 0 B3 5
```

In this case, the ".C" specifies that a clock pulse is to be applied to the CLK signal, the value 0 (low) to the CLR, the next sequential integer value to the 16 signals D0, D1, etc. and the hex value "5" (or 101 binary) to the signals FC2, FC1 and FC0. See definitions of these codes below.

The $INPUTS Command

All cells below this command in the same column are used to specify the input values to be set up in the circuit under test. As each line of the test program is read, the cell in this column is interpreted and new values are placed on the signal lines in the circuit. This command must be placed in the first row of this column to specify which signals are to be set up. The values specified in all subsequent rows are interpreted according to that header. See the information on header format above.

All non-command cells in this column are interpreted as signal value data. A data cell should contain one signal value item for each signal group specified in the most recent header (a signal group being either a single signal, or a group specified in brackets [] in the header. If the cell is empty or if less items are specified in the data cell than in the header, then unspecified items will be left unchanged from the previous time they were specified.

Each signal value item can be one of the items in the following table.

Value	**Comment**
A hexadecimal value	This will be interpreted as an integer with one bit for each signal in the corresponding header group. For individual signals, only the least significant bit is used, so the only values will be 0 or 1. For multiple-signal groups (i.e. specified in []'s in the header), the <u>last</u> signal in the group will take the value of the least significant bit and preceding signals will take the values of successively higher numbered bits. Unused high-order bits are ignored.
The value code ".C" or ".c"	This code indicates a positive clock pulse (i.e. a transition to the 1 level, followed by a transition to the 0 level), with a width as specified in the Configuration command. Note that if the signal is already at a 1 level, no rising transition will occur. For more information see the note below.

The value code ".N" or ".n"	This code indicates a negative clock pulse (i.e. a transition to the 0 level, followed by a transition to the 1 level), with a width as specified in the Test Options command. Note that if the signal is already at a 0 level, no falling transition will occur. See also the notes below.
The value code ".Z" or ".z"	This code indicates a high impedance value.
The value code ".X" or ".x"	This code indicates a “Don’t Know” value and can be used to test a circuits response to unknown inputs.
The increment operator "++" or "++n"	This code indicates that the integer value specified previously is to be incremented and the new value applied. If an integer is supplied immediately following the "++" then this number is added to the old input value, otherwise 1 is used.
The decrement operator "--" or "--n"	This code indicates that the integer value specified previously is to be decremented and the new value applied. If an integer is supplied immediately following the "--" then this number is subtracted from the old input value, otherwise 1 is used.

NOTE: Care must be taken regarding setup times when using the .C or .N clock pulse operators. E.g. .C generates a rising transition immediately which may not allow any setup time after values settled from the last step. This situation can be avoided in most cases by using .N to test positive-edge-triggered devices and .C for negative-edge-triggered devices. This ensures that a setup time at least the width of the clock pulse is allowed.

Command examples:

```
$I [D3..0]
$I CLK CLR LD/
$I FC0..2
```

The $EXPECTED Command

This command indicates that the following cells will contain the expected values for signals in the circuit under test. The values in the circuit are compared to those specified. The result of this comparison is shown as a mark character in each data cell. If the actual values equal the expected ones, then an "=" is displayed in the cell, otherwise a "#" is displayed. If no value appears in a data cell, then no comparison is done.

Each signal value item can be one of the items in the following table.

Value	Comment
A hexadecimal value	This will be interpreted as an integer with one bit for each signal in the corresponding header group. For individual signals, only the least significant bit is used, so the only values will be 0 or 1. For multiple-signal groups (i.e. specified in []'s in the header), the last signal in the group will take the value of the least significant bit and preceding signals will take the values of successively higher numbered bits. Unused high-order bits are ignored.
The value code ".X" or ".x"	This code indicates a "Don't Know" value and can be used to test a circuits response to unknown inputs
The value code ".Z" or ".z"	This code indicates a high impedance value.
The value code ".DC" or ".dc"	This code indicates a "Don't Care", i.e. it will always match. Note that this is different than not specifying a value, in that this will display an "=" result, whereas no comparison is done if no value is given.

The comparison is done at the same time step that the inputs are set or after waiting for the simulation to settle. See the menu command "" (page 120) for more information.

Command examples:

```
$E [ADDR0..15]
$E EQ
$E A B C D
```

The $OUTPUTS Command

This command specifies that the following cells should show the actual values read from the circuit and the result of the comparison with the expected values. Any cell in this column not containing a command will be overwritten when the outputs are read.

Command examples:

```
$O [ADDR0..15]
```

```
$O EQ
$O A B C D
```

The $TIME Command

This command specifies that the following cells should show the simulation time when execution of this step is started. Time values are displayed in decimal.

Command examples:

```
$TIME
```

The $WAIT Command

This command specifies that the following cells will contain signal value conditions to wait for before proceeding. This command is similar to $EXPECTED, except that instead of displaying an equal or non-equal status, the test program waits until an equal status occurs. Note the following rules:

> If the wait condition is already met, then the test program proceeds and the wait has no effect.
>
> The test program enters the wait state after any new input values specified in the row have been set up.
>
> If any delay is specified in the same row, then the delay is applied concurrently and it becomes, in effect, a minimum wait time.

Command examples:

```
$WAIT SYNC
$WAIT [D7..0]
```

The $DELAY Command

This command specifies that the following cells in the same column should be scanned for a decimal integer delay value. The following rules are applied:

> If a value is found, the specified delay is applied after setting up any input value specified in this row. I.e. This specifies the duration of any values applied on this line.
>
> If multiple $DELAYS appear in one row, then only the longest will apply.
>
> If no value appears in this column, then the default delay will apply.

Command examples:

```
$DELAY
```

NOTE: If any value appears after the $DELAY keyword, then it is taken to be a control command and the following cells in the column will not be interpreted as delays. See the description of this format below.

Control Commands

Command	**Short Form**	**Description**
$REPEAT or $REPEAT nnn	$R	Repeat rows up until the next $END either indefinitely or nnn times.
$END	$EN	Marks the end of a REPEAT loop.
$DELAY nnn	$D	Delay for nnn time units after applying the input values specified in this row.
$STOP	$S	Stop execution after this row.

$REPEAT Command

The $REPEAT/$END construct allows you to specify any integer number or an infinite repetition of a group of test vectors. The format of the $REPEAT command can be seen in the following example:

	$IN CLK	$IN CLR	$IN [D3..0]	$OUT [Q3..0]
	0	0	0	
	0	1	0	
	0	0	0	
$REP 32	0	0	++	
$END	1	0		

In this case, all vectors from the line containing the $REPEAT down to and including the line containing the $END will be repeated 32 times. If the repetition factor is omitted, it will be taken as infinite in interactive mode, and 1 in timing setup mode.

$REPEATs can be nested to any desired depth. If the $END is omitted, the last line in the data spreadsheet that has been modified is taken as the end of the loop.

$DELAY Command

When the $DELAY command appears with a value after it, it is taken as a control command and the specified delay is applied after execution of the current row. See the description of the $DELAY definition command above for more details.

Command example:

```
$D 15
```

$STOP Command

This command causes the test program to cease execution after applying any new input values specified in this row.

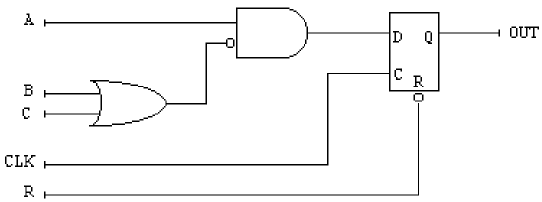
Test Vector Examples

This section provides some simple examples of Test Vector usage. For more complex examples, see the example files provided with the simulator release.

NOTE: Most of the devices in the Primitive device libraries contain stored test vectors and can be used in conjunction with the Place and Test Device command (page 126 in the Options menu for example purposes.

Testing a Circuit

In this example we will execute a set of test vectors for the following simple circuit made up of primitive devices:



Entering the Test Program

NOTE: This file is included with the Simulator release and can be loaded using the Open Test File command in the File menu.

Create a text file containing the following text. When creating this file, note these points:

You can create the file within DesignWorks using the built-in text editor by selecting the File/New command, selecting the Other Actions tab, then double-clicking on the Create a new, empty text document action. You can also use any other text editor, such as NotePad if you wish.

Save the file with the extension ".tsv".

Each row in the following table represents one line in the text file. Within each line there must be a tab character separating items. The actual layout of the text may not look exactly like what you see here, depending on the tab settings in the text editor.

$I R	$I CLK	$I C	$I B	$I A	$E OUT
0	0	1	1	0	0
1	0	1	1	0	0
1	.N	1	1	0	0
1	.N	0	1	0	0
1	.N	0	0	0	0
1	.N	0	0	1	1
1	.N	1	0	1	0

We are now ready to execute the test program.

Executing the Test Program

To execute the test program:

Display the Test Vector Panel, if it is not already, by selecting the Simulation/Test Vectors/Show Test Vector Panel menu command.

Click on the Run button and select the above test vector text file.

If all Test Vector and delay settings are the default values, this program will fail in the last step, due to a setup time problem. This will be explained in the next section.

Dealing With Delays and Setup Times

In this example, we are feeding inputs through a combinational logic array to an edge-triggered device. We must be aware that a delay must be allowed between changes at the combinational inputs and the application of the clock edge to the flip-flop.

In this case, the flip-flop is positive-edge-triggered. We have used the ".N" <u>negative</u> clock pulse notation since this in effect inserts a delay equal to the pulse width before the rising edge. This is because is creates a negative edge first (which is not significant to this flip-flop), then waits for the specified pulse width, then inserts the positive edge.

The default Test Vector clock pulse width is 2 units and the default delay in primitive devices is 1 unit. The last test vector in the above program causes a logic transition to pass through two gates, for a total delay of 2 units. Thus, the input transition at the D input to the flip-flop arrives at the same time as the positive edge of the clock. The D flip-flop model always uses the old value at the input, so the new value does not reach the Q output.

There are two solutions to this problem:

Change the Test Vector clock pulse width using the "Configuration" command in the Options menu. This will affect all clock pulses generated by this Test Vector window.

NOTE: These settings <u>are not</u> saved with the test file and will return to default values the next time the program is restarted.

Change the test program to set the clock input explicitly and insert a delay at this line, as in the following changes to the final line:

1	.N	0	0	1	1
$DEL 5	0	1	0	1	.DC
1	1	1	0	1	0

In this case, we are replacing the .N operator with two separate test vectors with explicit 0 and 1 values for the clock and a delay of 5 in between.

Primitive Devices

11

This chapter provides information on the "primitive" or built-in device types in DesignWorks. These types are intended primarily for use in creating model circuits for higher-level macro devices. Because their simulation functions are hard-coded, they occupy much less memory space than macro devices and simulate more efficiently.

NOTE:

1) In primitive devices, the association of logic functions to pins on a device symbol are made by pin order. When creating primitive devices using the DevEditor tool, you must be aware of the pin order requirements for the device type you are using. Refer to the description of each type in this chapter and to Appendix D - Primitive Device Pin Summary.

2) Bus pins are not supported on primitive device types.

The following table lists the available primitives and their functions.

Primitive Type	**Description**	**Variations using Same Primitive Type**	**Max. Size**
NOT	Inverter		1 input
AND	N-input AND gate	Any pin inversions	799 inputs
NAND	N-input NAND gate	Any pin inversions	799 inputs
OR	N-input OR gate	Any pin inversions	799 inputs
NOR	N-input NOR gate	Any pin inversions	799 inputs
XOR	N-input XOR gate		799 inputs
XNOR	N-input XOR gate		799 inputs
Transmission Gate	Transmission gate		1 bit
Three-State Buffer	Non-inverting N-bit 3-state buffer with optional common inverted enable	Buffer (non three state)	400 bits
Resistor	Digital resistor		
Multiplexer	M*N-to-M multiplexer		256 inputs
Decoder	1-to-N line decoder		256 inputs

Adder	N-bit adder with carry in and out	Incrementer	256 bits
Subtractor	N-bit subtractor with borrow in and out	Decrementer	256 bits
D Flip-Flop	D-type flip-flop	optional S & R	1 bit
JK Flip-Flop	JK flip-flop	T flip-flop, optional S & R	1 bit
Register	N-bit edge-triggered register		256 bits
Counter	N-bit synchronous counter	up/down	256 bits
Shift Register	N-bit shift register		256 bits
One Shot	Retriggerable one shot		
Clock	Clock oscillator		
Binary Switch	Debounced toggle switch		
SPST Switch	Open/closed single pole switch		
SPDT Switch	Double throw switch		
Binary Probe	Signal level display		
Hex Keyboard	Hexadecimal input device		
Hex Display	Hexadecimal digit display		
SetupHold	Setup/Hold time checker		
Unknown	Unknown value detector		

The following table lists devices supported primarily for compatibility with older version of DesignWorks. We do not recommend using these in new designs.

Device	Description
Pullup	Pullup resistor, single pin
D Flip-Flop ni	D-type flip-flop (non-inv S & R)
JK Flip-Flop ni	JK flip-flop (non-inv S & R)
Glitch	Glitch detector
SimStop	Simulation halt device

Gates and Buffers

The Primitive Gates library contains the primitive gates with a built-in simulation function. The NOT, AND, NAND, OR, NOR, XOR and XNOR devices behave according to the appropriate truth tables for such devices. Any gate input which is in the “Don’t Know”, “High Impedance” or “Conflict” state is treated as a “Don’t Know”. A gate with a “Don’t Know” input will not necessarily produce a “Don’t Know” output. For example, if one input on an AND gate is low, the output will be low, regardless of the state of the other input, as in the following truth table.

Gate Definition

The gate types, except NOT, can be created with any number of inputs (from 0 to 799) and are defined as shown in the following table.

Function	**Output is...**	**Output is DONT if...**
AND	LOW if any input is LOW, HIGH otherwise	Some input is DONT and no input is LOW
NAND	HIGH if any input is LOW, LOW otherwise	Some input is DONT and no input is LOW
OR	HIGH if any input is HIGH, LOW otherwise	Some input is DONT and no input is HIGH
NOR	LOW if any input is HIGH, HIGH otherwise	Some input is DONT and no input is HIGH
XOR	HIGH if an odd number of HIGH inputs and no DONTs	Any input is DONT
XNOR	HIGH if an even number (or zero) of HIGH inputs and no DONTs	Any input is DONT

Gate Pin Order

The NOT type must have exactly one input and one output, in that order. All other logic gate types can have any number of inputs followed by a single output, up to the maximum DesignWorks limit of 800 pins.

NOTE: Pin order is important in all primitive devices! When creating a gate type using the DevEditor tool , the output pin must be the last item on the pin list.

Pin Inversions

The logic of any pin on any device can be inverted by placing a non-empty value in the Invert.Pin attribute field of the pin.

For example, to create the following AND gate with one inverted pins:

the following steps must be taken in the symbol editor tool:

1) Create the desired graphic symbol using the DevEditor's drawing tools.

2) Place the three pins as shown. ORDER IS IMPORTANT! All primitive devices must have a specific pin order. For gates, all inputs come first and the output pin last.

3) In the pin list, double-click on the last pin (the output pin). This will display the Pin Info palette for that pin.

4) Set the pin type to Output, and, if desired, edit the pin name.

5) Press the Enter key on the keyboard to move to the next pin and edit the other pin names, if desired, and check that they are all set to Input.

6) Click the close box in the Pin Info palette.

7) In the pin list, click once to select the input pin that is to be inverted, then select the Pin Attributes command in the Options menu.

8) Select the All Fields option in the attributes dialog then select the Invert.Pin field.

9) Enter the value "1" for this field, then click Done. (The actual value doesn't matter, as long as it is non-empty.)

10) Select the Sub-circuit / Part Type command in the Options menu

11) Click on the Primitive Type button, then select the AND primitive type in the menu.

12) Close the Part Type dialog and save the part to a library in the usual manner.

NOTE:

1) The logical inversion of the pin is completely independent of the graphical representation of the pin. I.e. using the "inverted pin" graphic in the DevEditor tool does not invert the pin logic in the simulator. You must set the Invert.Pin field to have this effect.
2) Inverted gate types NAND and NOR can be created by using the NAND and NOR primitive type setting or by using the AND and OR settings and inverting the output pin (or the input pins, using DeMorgan's Theorem). The two methods will produce identical simulation results. There is a slight memory overhead but no execution speed overhead to using an inverted pin.

Transmission Gate

The transmission gate (X-Gate) device behaves as an electrically controlled SPST switch. When the control input is high, any level change occurring on one signal pin will be passed through to the other. Since it has no drive capability of its own it will behave differently than a typical logic device when a high-impedance or low drive level signal is applied to its signal inputs. Most other primitives, such as gates, interpret any applied input as either High, Low or Don't Know. The transmission gate, on the other hand, will pass through exactly the drive level found on its opposite pin. Thus, a high-impedance level on one pin will be transmitted as a high-impedance level on the other pin. Note that the simulation of this device may produce unpredictable results in extreme cases, such as an unbroken ring of transmission gates.

NOTE: No variations in number or order of pins are possible with the XGATE primitive type. It must have exactly one control pin and two bidirectional pins with pin order as described in Appendix D - Primitive Device Pin Summary.

Three-State Buffer

The three-state buffer has N data inputs, N data outputs, and an optional active-low enable input. If the enable input exists and is high, all outputs enter a "High-impedance" state. If the enable input doesn't exist, or is low, each output will follow the corresponding input if it is low or high, or produce a "Don't Know" level otherwise.

A single-input three-state buffer is shown in the following table:

Enable	Data	Out
0	0	0
0	1	1
1	0	Z

1	1	Z

Making Non-Inverting Buffers

The Buffer primitive type can also be used to make a non-inverting buffer, i.e. with its outputs always enabled, simply by omitting the enable input.

This can be used for the following purposes:

To represent a non-inverting buffer or level translator in a design.

To insert a delay in a signal path without affecting the logic of the signal.

To create various types of open collector, open emitter or inverting buffers, when used in conjunction with different pin type and inversion settings on the outputs. NOTE: It is more efficient to use the NOT primitive type to make a simple inverter.

Resistor

The resistor device simulates the effects of a resistor in a digital circuit. It is more general than the Pullup Resistor device and can be used as a pullup, pulldown or series resistor. Whenever a signal level change occurs on either pin of the resistor, the device converts that level into a resistive drive level (see (page 13) for more information on drive levels). A high impedance drive on one end is transmitted as a high-impedance output to the other end. Note that DesignWorks does not simulate analog properties of devices so the resistor device does not have a resistance value in the analog sense and will not interact with capacitor symbols placed on the same line. The effect of resistance on line delay can be simulated by setting the delay of the resistor device.

Logic Devices

Multiplexer

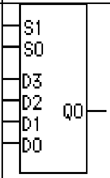
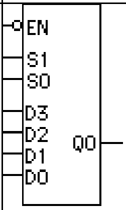
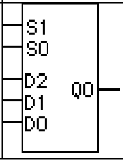
This is a device which selects one of N data inputs and routes it to a corresponding output line. There can be from 1 to 256 outputs, plus an optional enable input, as long as the total pin count does not exceed the 800 pin limit.

A typical 8-to-1 multiplexer obeys the following function table. X = Don't Care.

EN	**S2**	**S1**	**S0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**G**
0	0	0	0	X	X	X	X	X	X	X	0	0
0	0	0	0	X	X	X	X	X	X	X	1	1
0	0	0	1	X	X	X	X	X	X	0	X	0
0	0	0	1	X	X	X	X	X	X	1	X	1
0	0	1	0	X	X	X	X	X	0	X	X	0
0	0	1	0	X	X	X	X	X	1	X	X	1
0	0	1	1	X	X	X	X	0	X	X	X	0
0	0	1	1	X	X	X	X	1	X	X	X	1
0	1	0	0	X	X	X	0	X	X	X	X	0
0	1	0	0	X	X	X	1	X	X	X	X	1
0	1	0	1	X	X	0	X	X	X	X	X	0
0	1	0	1	X	X	1	X	X	X	X	X	1
0	1	1	0	X	0	X	X	X	X	X	X	0
0	1	1	0	X	1	X	X	X	X	X	X	1
0	1	1	1	0	X	X	X	X	X	X	X	0
0	1	1	1	1	X	X	X	X	X	X	X	1
1	X	X	X	X	X	X	X	X	X	X	X	1

Multiplexer Pin Variations

A number of variations in multiplexer logic are possible with this primitive type, depending on which input and output pins are included. The following table summarizes the possible variations. Samples are shown with M=1 and N=2, but any combination of M and N can be used within the maximum pin limit of 800.

Number of Sections	Number of Inputs/Section	Number of Select Inputs	Number of Enable Inputs	Sample Symbol
M	2^N	N	0	
M	2^N	N	1	
M	$2^{N-1}+1 .. 2^N$	N	0*	

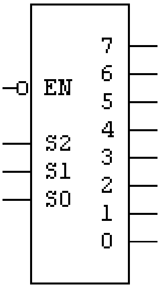
* If there are less than 2^N inputs per section there can be no enable input.

See "Pin Function Table" on page 175 for specific pin order requirements for multiplexer types.

Decoder

The Decoder activates one of N outputs, depending on M select inputs, as follows (X = Don't Care):

EN	**S2**	**S1**	**S0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	1	1
1	X	X	X	1	1	1	1	1	1	1	1



Adder/Incrementer

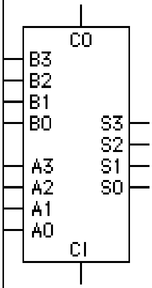
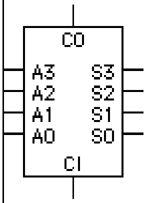
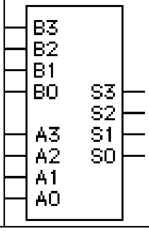
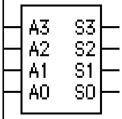
The N-bit Adder accepts one or two N-bit input arguments and (optionally) a 1-bit carry and outputs their N-bit sum plus an optional 1-bit carry out.

Multiple adders can be connected together by feeding the carry out from each stage to the carry in of the next more significant stage. The carry in to the least significant stage should be set to zero.

Adder Pin Variations

The adder primitive can be used in four variations, as summarized in the following table. Sample symbols are shown with 4 bit inputs, although any number of bits from 1 to 256 is permissible.

	Has B Inputs	No B Inputs

Has Carry In	S = A + B + Cin  CO B3 B2 B1 B0 S3 S2 S1 S0 A3 A2 A1 A0 CI	S = A + Cin 
No Carry In	S = A + B  B3 B2 B1 B0 S3 S2 A3 S1 A2 S0 A1 A0	S = A + 1 

In addition, the Carry Out pin can be independently included or omitted from any of these configurations.

See "Pin Function Table" on page 175 for specific pin order requirements for adder/incrementer types.

Subtracter/Decrementer

The Subtracter primitive type behaves identically to the Adder type except that a subtract or decrement operation is performed, depending upon pin configuration.

D Flip-Flop

The D-type flip-flop is positive-edge triggered and obeys the following function table:

S	**R**	**D**	**Clock**	**Q**	**Q/**
0	0	X	X	1	1
0	1	X	X	1	0
1	0	X	X	0	1
1	1	0	Rises	0	1

1	1	1	Rises	1	0
Rises	Rises	X	X	X	X

In the above table X on the input side means "Don't Care" and on the output side means "Don't Know".

Flip-Flop Setup and Hold Times

None of the DesignWorks primitive types explicitly implement variable setup and hold times. However, all edge-triggered primitives have an effective setup time of 1 unit since they always use the input signal value existing before the current step. I.e. if the data input changes at the same time as the clock, the old data value will be used to determine the new output value. This effective setup time can be modified by specifying input pin delays on either the data or clock pins.

Checking for setup and hold violations can be done in one of two ways:

Attaching a SetupHold primitive device to the inputs and output of the device under test. The combined device could be made into a sub-circuit if desired to combine it into a single symbol.

Using the Trigger capability of the simulator to watch for value changes within a set amount of time.

Flip-Flop Initialization

Note that when a flip-flop is first placed in the schematic, it is in an unknown state and must be correctly initialized before it will produce predictable outputs. This can be done in the following ways:

Adding circuitry to force an explicit reset.

Using the Clear Unknowns command in the Simulator window to force an initial state before starting the simulation.

Specifying an initial output value for both the Q and Q/ outputs in their respective Initial.Pin attributes. This will be applied every time a Clear Simulation command is executed.

D Flip-Flop Optional Pins

The D Flip-Flop primitive type has the following optional pins:

The Q/ (Not-Q) output can always be omitted.

The Set input alone or both the Set and Clear inputs can be omitted.

See “Pin Function Table” on page 175 for specific pin order requirements for D Flip Flop types.

D Latch

The D Latch primitive type is identical to the D Flip-Flop in function and pin specifications except that it is level-triggered instead of edge-triggered. I.e. The Q and Q/ outputs will follow the level of the D input as long as C is high.

JK Flip-Flop

The JK flip-flop is negative-edge triggered and obeys the following function table:

S	R	J	K	Clock	Old Q	New Q	New Q/
0	0	X	X	X	X	1	1
0	1	X	X	X	X	1	0
1	0	X	X	X	X	0	1
1	1	0	0	falls	0	0	1
1	1	0	0	falls	1	1	0
1	1	0	1	falls	X	0	1
1	1	1	0	falls	X	1	0
1	1	1	1	falls	0	1	0
1	1	1	1	falls	1	0	1
rises	rises	X	X	X	X	X	X

In the above table X on the input side means "Don't Care" and on the output side means "Don't Know".

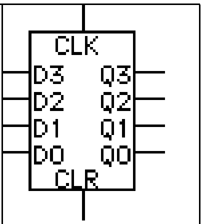
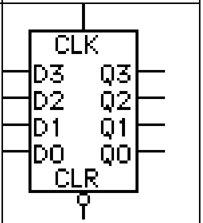
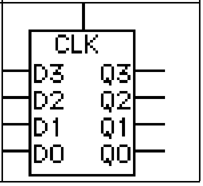
If any inputs are in an unknown state, the simulator will determine the output state where possible, or else set it to "Don't Know".

See “Pin Function Table” on page 175 for specific pin order requirements for D Flip Flop types.

Register

This device implements an N-bit, positive-edge-triggered register with common clock and an optional active-high clear inputs.

The following table illustrates some pin variations available for the Register primitive type.

4-bit register with active-high clear	
4-bit register with active-low clear (using pin inversion)	
4-bit register without clear	

See "Pin Function Table" on page 175 for specific pin order requirements for D Flip Flop types.

Counter

This device implements a N-bit, presettable, synchronous, positive-edge-triggered up/down counter with active-low enable. The load data inputs and most of the control inputs can be omitted for simplified versions.

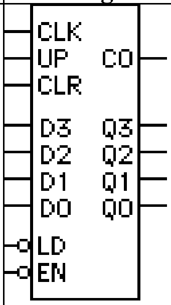
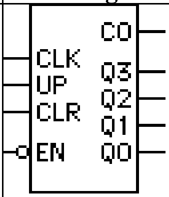
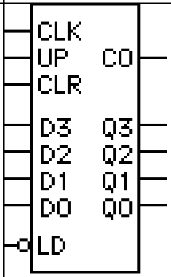
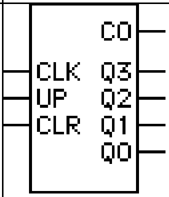
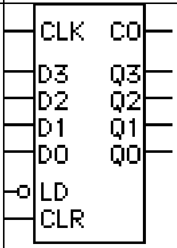
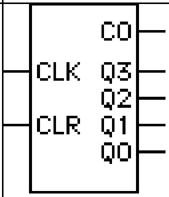
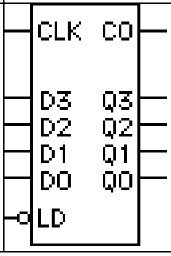
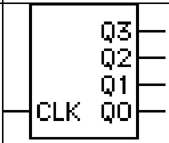
The following timing diagram shows a typical count cycle. Note that the CO (Carry Out) output goes low when the count reaches 2^N-1 (when counting up) or 0 (when counting down) and rises again on the next count. This can be used to cascade multiple counters together, as shown. The CLR input clears the counter asynchronously (i.e. regardless of the state of the clock). The Count/Load input, when low, causes the data from the N data inputs to be passed to the outputs on the rising edge of the next clock. The Enable input disables counting when high, but has no effect on loading.

Cascading Multiple Counters

Counter primitives with the optional Enable and Carry Out pins can be cascaded to form larger synchronous counters as follows:

Counter Pin Variations

The following table summarizes the possible pin usage variations for the counter primitive type. The sample are shown with N=4, although the number of bits can be anywhere in the range 1 to 256.

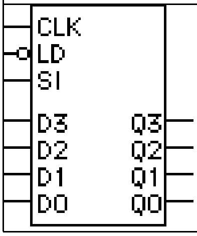
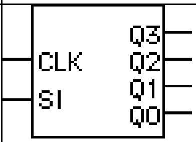
Optional Inputs	Including Load Inputs	Excluding Load Inputs
CLR, UP/DN, ENABLE	 CLK UP CO CLR D3 Q3 D2 Q2 D1 Q1 D0 Q0 LD EN	
CLR, UP/DN	 CLK UP CO CLR D3 Q3 D2 Q2 D1 Q1 D0 Q0 LD	
CLR	 CLK CO D3 Q3 D2 Q2 D1 Q1 D0 Q0 LD CLR	
none	 CLK CO D3 Q3 D2 Q2 D1 Q1 D0 Q0 LD	

NOTE: CO can be independently included or omitted in any of the above variations.

Shift Register

The shift register is an N-bit, positive-edge-triggered device with serial or optional parallel load. When the Shift/Load input is low, data from the N parallel data input lines is transferred to the outputs on the rising edge of the next clock. When Shift/Load is high, the next rising clock edge causes the value at the Shift In input (SI) becomes the new value for output Q0, Q0 shifts to Q1, Q1 to Q2, etc., and the old value at the most significant output is lost.

The following table shows the shift register primitive with and without parallel inputs.

With Parallel Load	Without Parallel Load
 CLK LD SI D3 D2 D1 D0 Q3 Q2 Q1 Q0	

NOTE: The Shift Register primitive cannot be created without data outputs (i.e. a parallel-in serial-out register) because the flip-flop values are stored on the output pins. Primitive devices have no internal state storage. See (page 13) for more information.

Clock

The clock oscillator is used to generate a repeating signal to activate other devices. When it is first created, the clock output pin will be low, then after a delay time called the "Low Time" it will change to the high state. After a further delay called the "High Time" the signal will revert to low, and the cycle will repeat. The low and high times are initially set to 10, but can be modified using the Parameters command in the Options menu in the Simulator window. Any number of Clocks may exist at once with independent delay times.

Creating Synchronized or Offset Clocks

When the Clear Simulation command is selected (or the Restart button on the Timing palette pressed), all clocks in the design are restarted. Clock outputs will be set to the low state and the timer for the low period will be restarted. Clock high and low times, combined with pin inversion and pin delay settings can be used to precisely determine the relationship between two clock outputs. The following circuit example summarizes these options.

Signal	Low Time	High Time	Invert.Pin	Pin Delay
CLK	10	10		0
CLKx2	5	5		0
CLK.INV	10	10	1	0
CLK.DELAY	10	10		5
CLK.INV.DELAY	10	10	1	5

Setting Clock Values

To set the high and low times for a clock, first select the device in question (by setting the cursor to "Point" mode and clicking the mouse button inside the device symbol), then choose the Parameters command in the Options menu in the Simulator window.

You will be presented with a dialog containing buttons for increasing or decreasing the high and low values. The minimum for either value is 1 and the maximum is 32767.

See “” (page 47) in for more information.

One Shot

The one shot is used to generate an output pulse of a fixed length when it is triggered by the rising edge of the trigger input. Two parameters can be set for a one shot:

the delay from the rising edge of the input to the start of the output pulse.

the duration of the pulse.

The delay and duration times are initially set to 1 and 10, respectively, but can be modified using Parameters command in the Options menu in the Simulator window.

The One Shot device is retriggerable, meaning that the output pulse will not end until "duration" time units has passed since the last trigger input. Repeating the trigger input can cause the output pulse to be extended indefinitely.

I/O Simulation Pseudo-Devices

Binary Switch

The Binary Switch device allows setting of a signal to a low or high level. When a switch is first created its output is at a low level. Pointing at the switch with the cursor in "Point" mode and clicking the mouse button causes the switch arm to move and the output to change to the opposite state. Any number of device inputs can be driven by a switch output. A switch has no delay characteristic since it has no inputs. The switch can be selected by holding the shift key down and clicking on the switch device.

SPST Switch

The SPST switch device simulates the actions of a simple open/closed switch in a digital circuit. When a switch is first created it is open and both connections present a high impedance logic level. Clicking on the switch with the cursor in "Point" mode causes the switch arm to close and the switch to "conduct". In terms of the digital simulation, this means that whatever logic level is present on each pin is transmitted to the other one.

An SPST switch has a default delay of zero but this can be set to any value using the Parameters command in the Options menu in the Simulator window. The switch can be selected by holding the shift key down and clicking on the switch device.

SPDT Switch

The SPDT switch device operates in essentially the same manner as the SPST switch described above, except that it is always conducts between the single pin on one side and one of the two pins on the other. As with the other two switch types, clicking on it with the Point cursor changes the position of the contact. The switch can be selected by holding the shift key down and clicking on the switch device.

Binary Probe

The binary probe is a device for displaying the level present on any signal line. When the probe is first created, its input is unconnected and therefore in the "High-impedance" state, which will be displayed as a "Z". When the input pin is connected to another signal, the displayed character will change to reflect the new signal's current state. Any further changes in the signal state will be shown on the probe.   Possible displayed values are 0 (low), 1 (high), X (Don’t Know), Z (High Impedance) or C (Conflict).

Hex Keyboard

The hex keyboard outputs the binary equivalent of a hexadecimal digit on four binary lines. A "key" is pressed by positioning the tip of the arrow cursor in the desired key number and clicking the mouse button. The binary data on the output lines will change to reflect the new value and remain until the next key is pressed. On a fifth line, it outputs a signal which goes high momentarily and then low again when a key is pressed. The hex keyboard can be selected by holding the shift key down and clicking on the hex keyboard.

Hex Display

The hex display shows the hexadecimal equivalent of its four binary inputs. If any of the inputs is unknown, high-impedance or conflict, then an X will be displayed.

SetupHold

Setup and hold times can be checked by attaching a "SetupHold" primitive device to the inputs of the clocked device to be checked, as follows:

Each time the "D" input changes, the SetupHold device marks the current time. On the next rising edge of the "C" input, the current time is compared to the last transition on the "D" input. If the difference is less than or equal to the Setup time specified in the Simulation Parameters dialog, the output switches to a Don't Know state and remains in this state until a non-violating input transition occurs. Similarly, for each clock rising edge, the time is marked and compared to the time at the next transition on the "D" input. If this difference is less than the Hold time given, the output is set to Don't Know.

The SetupHold device puts out a highZ value until a setup or hold violation occurs when it switches to a don't know state. Thus, the output of the SetupHold device can be paralleled with the flip-flop so that the output line will enter a conflict state when an error occurs.

Unknown Detector

The Unknown Detector device detects any unknown value (i.e. not 0 or 1) on its input and puts out a high value after the delay time specified in the Simulation Parameters dialog. When the input returns to a 0 or 1 state, the output becomes zero. This can be used in conjunction with triggers to detect invalid conditions in a circuit and stop the simulator.

The PROM/PLD/ RAM Tool

12

Introduction

TheThe Prom/Pld/Ram Tool is used to create and view custom memory devices employing user-specified data. The following devices can be built:

RAM (Random Access Memory)

PROM (Programmable Read Only Memory)

PLA (Programmable Logic Arrays)

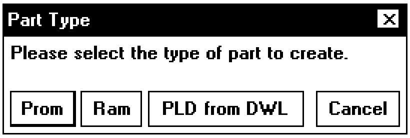
PLD (Programmable Logic Devices)

Starting the Prom/Pld/Ram Tool

To start the Prom/Pld/Ram tool, select the PROM/RAM/PLA Wizard command in the Simulaiton menu.

Prom/Pld/Ram Window

The Prom/Pld/Ram window is displayed when the Prom/Pld/Ram tool is selected from the “Tool” menu in the DesignWorks window and no PROM, PLD, PLA or RAM devices are currently selected in the active schematic window. The window can be hidden by selecting “Close” from the system menu in the top left corner of the window.



Part Type

Please select the type of part to create.

Prom/Pld/Ram Controls and Fields

Control	Comment
Prom	Build a custom PROM device.
Ram	Build a custom RAM device.
PLD from DWL	Build a PLD from DWL file.

PROM Device

The PROM primitive device supports the direct simulation of Programmable Read Only Memory devices in a variety of configurations. This allows these devices to be efficiently represented as a single simulation device, rather than having to generate a netlist of equivalent logic devices. This tool allows creation of custom PROM devices with a variety of options.

Description of the PROM Primitive

For the purposes of simulation in DesignWorks, a PROM (Programmable Read Only Memory) is defined as a device having N inputs, M outputs and having 2N storage locations, each containing M bits. Each different input combination selects one of the storage locations, the contents of which appear on the output lines. The number of storage locations required doubles for each input bit added, so PROM organization is only practical for a relatively small number of inputs. The advantage of the PROM is that any arbitrary Boolean function can be represented simply by storing the truth table for the function in the appropriate storage locations.

PROM Limitations

PROM devices must fall within all of the following limits:

a maximum of 30 address line inputs.

a maximum of 256 bits per word.

total PROM memory space must be less than 231 bytes.

sufficient free memory in the system to allocate a block twice the size of the PROM memory space.

Creating PROM Devices

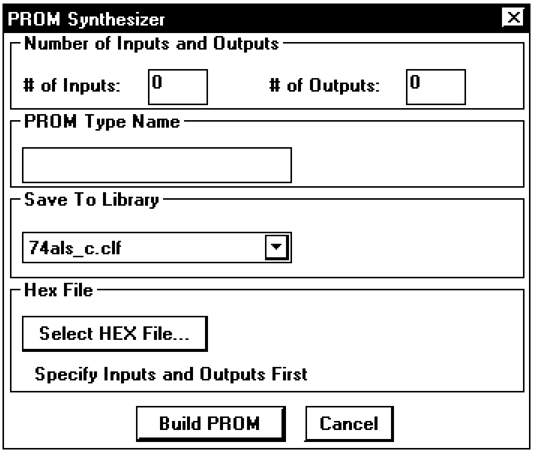
No manual method for entering data directly into a PROM device exists. The internal data must be read from a text file in a specific format. These files are intended to be machine-generated using a program such as the MacABEL™ or student ABEL™ logic compilers. However, the specified hex file format used for loading PROM data does not include any information on device structure, but only a raw listing of binary data. For this reason, this tool requires the structure to be defined manually while the internal data is read from a file.

Hex Data File

The hex file used for PROM loading must be in the commonly used Intel MDS hex format. This format can be generated by the MacABEL™ or student ABEL™ logic compilers and by a variety of microprocessor assemblers and compilers. The file format is described in the appendix at the back of this manual.

Defining the Structure

A custom PROM device can be created using the Prom/Pld/Ram tool and selecting the PROM control. The following dialog is displayed:



PROM Synthesizer

Number of Inputs and Outputs

# of Inputs: 0

# of Outputs: 0

PROM Type Name

Save To Library: 74als_c.clf

Hex File

Specify Inputs and Outputs First

In order to create the PROM device, the format of the device (i.e. number of inputs and outputs) must be specified along with the hex file to be read. The following table summarizes the controls in this dialog:

Control	**Comment**
# of Inputs	The number of address inputs.
# of Outputs	The number of data outputs, i.e. the number of bits per storage location.
PROM Type Name	The name of the part as it will appear in the parts palette and on the symbol.
Save To Library	This pop-up menu allows you to select the destination library for the completed part.
Select HEX File...	This button allows you to select the hex data file. By default, only files with the extension ".hex" are shown. A button on this file box allows you to display all text files. The format of this file is described in a technical note supplied with this manual.
Hex Filename	This shows the currently-selected input file. This is set using the Select HEX File button.

Editing PROM Devices

There is no way to redisplay the PROM dialog in order to re-specify build parameters for an existing device. Once a PROM device definition has been placed into a library, only limited editing changes can be made using the

DevEditor tool.

NOTE: NOTE: The PROM device definition contains structure information that <u>cannot</u> be edited after the device is created. Adding or deleting any pins using the DevEditor will invalidate the device definition and render it useless.

The DevEditor can be used to make the following changes to a PROM device, if desired:

any graphical changes to the symbol.

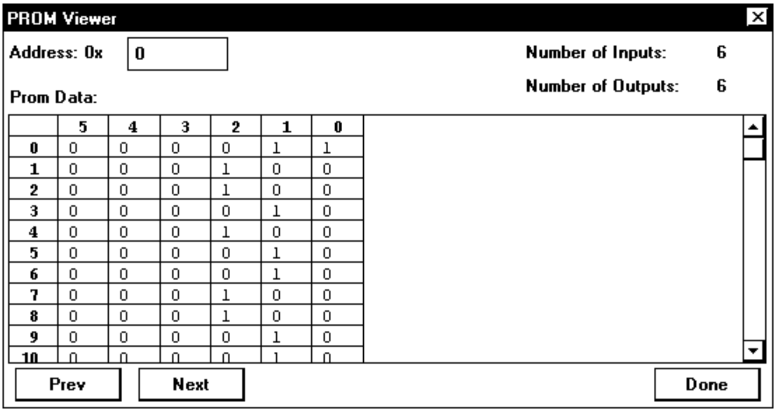
pin name, visible pin number or pin attribute changes (including pin delay and inversion).

limited pin type changes (e.g. changing to open collector).

part attribute changes.

Viewing PROM Device Contents

The contents of an existing PROM device on a schematic can be viewed (but not changed) by selecting the device on the schematic, then selecting the Prom/Pld/Ram tool from the “Tools” menu in the DesignWorks window. The following dialog is displayed:



Address: 0x 0

Number of Inputs: 6

Number of Outputs: 6

Prom Data:

	5	4	3	2	1	0
0	0	0	0	0	1	1
1	0	0	0	1	0	0
2	0	0	0	1	0	0
3	0	0	0	0	1	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	1	0
7	0	0	0	1	0	0
8	0	0	0	1	0	0
9	0	0	0	0	1	0
10	0	0	0	0	1	0

The following table summarizes the controls in this dialog:

Control	Comment
Address	Enter the address to be displayed. If the address entered is greater than the number of addresses for the device then the last eight address will be displayed.

Next	Display the next 16 addresses in the dialog
Prev	Display the previous 16 addresses in the dialog

Ram Device

The RAM primitive device supports the direct simulation of static Random Access Memory devices in a variety of configurations. This allows these devices to be efficiently represented as a single simulation device, rather than having to generate a netlist of equivalent logic devices. This tool allows creation of custom RAM devices with a variety of pin options.

Description of the RAM Primitive

The following table summarizes the options available in the RAM primitive type:

Option	Comment
Chip Enables	0, 1, 2 or 3 active-low chip enables. If any chip enable input is high, all read and write functions are disabled. In no enable is provided, the device will always be enabled.
Write Enable	The active-low Write Enable pin is not optional. A low level on this pin causes the data present at the Data In lines to be written to the location selected by the address lines.
Output Enable Pin	This active-low pin controls output enable but does not affect writing.
Data In/Out	The data input and output lines can either be separate or combined into a single I/O bus.
Three-state outputs	If the input and output lines are combined, or if three-state outputs are specified, the outputs enter a high-impedance state if Write Enable or any Chip Enable are high. If three-state outputs are not specified, data outputs will be high when disabled.

Single-word Simulation	If this option is selected, only a single word of real memory will be allocated for simulation purposes, i.e. the address inputs will be ignored. This allows logic testing of a circuit containing a large RAM device without consuming large amounts of program memory.
Common I/O	This option specifies that a single I/O pin will be used per data bit, rather than separate data in and data out lines. In this case, three-state outputs are assumed and outputs will be disabled when writing.

<u>RAM Pin Options</u>

The normal pin delay (using the Delay.Pin attribute field) and pin inversion (using the Invert.Pin attribute field) options can be used with RAM devices.

See the Simulator and Associated Tools manual, Chapter III - Simulation Details for more information on pin delay and inversion.

<u>RAM Limitations</u>

Custom RAM devices must meet all of the following limits:

a maximum of 30 address line inputs.

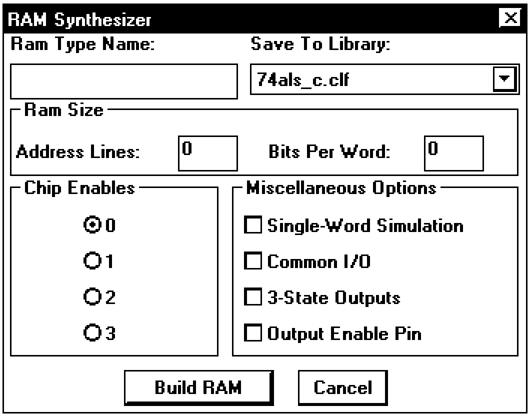
a maximum of 256 bits per word.

total RAM memory space must be less than 2^{31} bytes (even if single word simulation is used).

sufficient free memory in the system to allocate a block twice the size of the RAM memory space, unless single word simulation is used.

Creating RAM Devices

A custom RAM device can be created using the Prom/Pld/Ram tool and selecting the RAM control. The following dialog is displayed:



RAM Synthesizer

Ram Type Name:

Save To Library: 74als_c.clf

Ram Size

Address Lines: 0

Bits Per Word: 0

Chip Enables

● 0

○ 1

○ 2

○ 3

Miscellaneous Options

□ Single-Word Simulation

□ Common I/O

□ 3-State Outputs

□ Output Enable Pin

NOTE: NOTE: The RAM tool will require you to select a library to which the new device definition will be saved. You may wish to create a library for this purpose (using the pop-up menu in the parts palette) before proceeding.

The following table summarizes the controls in this dialog:

Control	**Comment**
Type Name	The device will be saved to the library under this name.
Save To Library	This pop-up menu allows you to select an open library to save the device to.
Address Lines	This specifies the number of address lines for the device, in the range 1 to 30
Bits Per Word	This specifies the number of bits (1 to 256) per memory location, which will also be the number of data input pins and data output pins.
Chip Enables	These buttons allow you to select 0, 1, 2 or 3 chip enable inputs.
Single-word Simulation	This option enables single-word simulation as described in the previous section.
Common I/O	This option specifies that a single I/O pin will be used per data bit, rather than separate data in and data out lines.
Three-state outputs	This option enables three state outputs as described in the previous section.
Output Enable Pin	This will add an additional control pin which controls output enable but does not affect writing.

Editing RAM Devices

There is no way to redisplay the RAM dialog in order to re-specify build parameters for an existing device. Once a RAM device definition has been placed into a library, only limited editing changes can be made using the DevEditor tool.

NOTE: NOTE: The RAM device definition contains structure information that <u>cannot</u> be edited after the device is created. Adding or deleting any pins using the DevEditor will invalidate the device definition and render it useless.

The DevEditor can be used to make the following changes to a RAM device, if desired:

any graphical changes to the symbol.

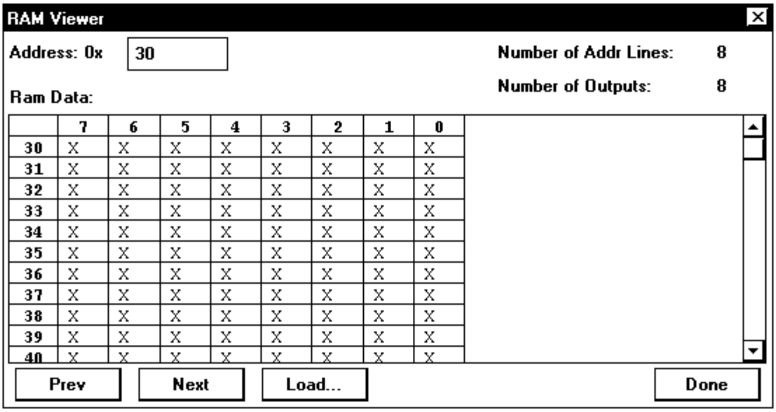
pin name, visible pin number or pin attribute changes (including pin delay and inversion).

limited pin type changes (e.g. changing to open collector).

part attribute changes.

Viewing RAM Device Contents

The contents of an existing RAM device on a schematic can be viewed (but not changed) by selecting the device on the schematic, then selecting the Prom/Pld/Ram tool from the “Tools” menu in the DesignWorks window. The following dialog is displayed:



Address: 0x 30

Number of Addr Lines: 8

Number of Outputs: 8

Ram Data:

	7	6	5	4	3	2	1	0
30	X	X	X	X	X	X	X	X
31	X	X	X	X	X	X	X	X
32	X	X	X	X	X	X	X	X
33	X	X	X	X	X	X	X	X
34	X	X	X	X	X	X	X	X
35	X	X	X	X	X	X	X	X
36	X	X	X	X	X	X	X	X
37	X	X	X	X	X	X	X	X
38	X	X	X	X	X	X	X	X
39	X	X	X	X	X	X	X	X
40	X	X	X	X	X	X	X	X

The following table summarizes the controls in this dialog:

Control	Comment
Next	Display the next 16 addresses in the dialog
Prev	Display the previous 16 addresses in the dialog
Load	This option has not yet been implemented.
Address	Enter the address to be displayed. If the address entered is greater than the number of addresses for the device then the last eight address will be displayed.

PLA Device

The PROM primitive device supports the direct simulation of Programmable Logic Array devices in a variety of configurations. This allows these devices to be efficiently represented as a single simulation device, rather than having to generate a netlist of equivalent logic devices.

Description of the PLA Primitive

In DesignWorks, a PLA (Programmable Logic Array) consists (in effect) of a group of AND gates feeding into a single OR (active high) or NOR (active low) gate for each output bit. Each AND-gate input is connected to either an input bit, the inverse of an input bit, or constant high. By selectively making these input connections it is possible to determine which input combinations will produce 0's or 1's in the outputs. PLAs are actually represented internally in a compact binary format, not as a netlist of AND and OR gates.

The input connections required to implement simple logic functions can generally be determined "by eye" for simple cases, whereas more complex logic must be reduced using Karnaugh maps, the Quine-McClusky method or other more advanced design techniques. These methods are discussed in numerous circuit design textbooks and will not be covered here. DesignWorks has the capability of reading device data produced by external logic compiler programs such as the MacABEL™ or student ABEL™ logic compilers.

PLA Limitations

PLA devices must fall within the following limits:

the number of inputs plus number of outputs must be less than 801.

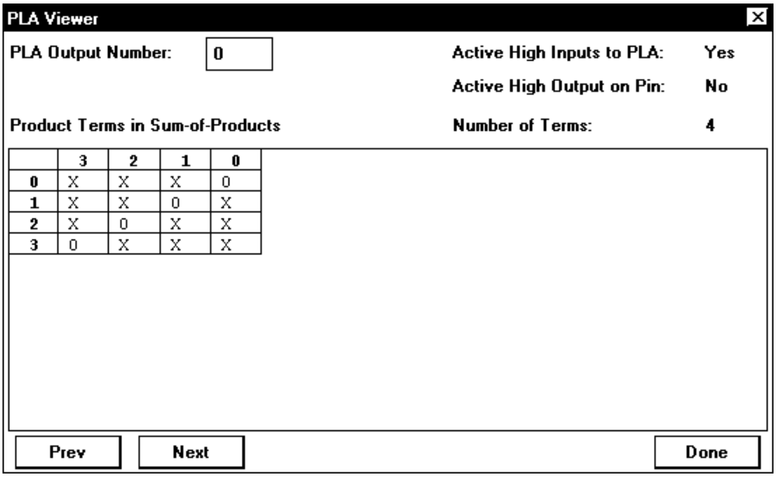
the number of product terms per output must be less than 65536.

Creating PLA Devices

There is no direct way to produce PLA devices. They are created as a component of a PLD device. See the section on PLD devices for more information.

Viewing PLA Device Contents

The contents of an existing PLA device on a schematic can be viewed (but not changed) by selecting the device on the schematic, then selecting the Prom/Pld/Ram tool from the "Tools" menu in the DesignWorks window. The following dialog is displayed:



PLA Viewer

PLA Output Number: 0

Active High Inputs to PLA: Yes

Active High Output on Pin: No

Product Terms in Sum-of-Products

Number of Terms: 4

	3	2	1	0
0	X	X	X	0
1	X	X	0	X
2	X	0	X	X
3	0	X	X	X

The following table summarizes the controls in this dialog:

Control	Comment
Next	Display the next 16 addresses in the dialog
Prev	Display the previous 16 addresses in the dialog

Address	Enter the address to be displayed. If the address entered is greater than the number of addresses for the device then the last eight address will be displayed.

PROM Data File Format

The file will contain one byte per PROM address if 8 or less output bits were specified, two bytes per PROM address if 9 to 16 output bits were specified, etc. The number of addresses in the file will be 2^N, where N is the number of input bits. Each line must start with a colon ":", followed by:

2 hex digits (1 byte) indicating the number of actual data bytes on this line.

4 hex digits (2 bytes) indicating the starting address for the data bytes on this line.

2 hex digits which will be "00" in all records except for an empty terminator record, in which they will be "01".

groups of 2 digits for the number of data bytes indicated at the beginning of the line.

2 hex digits for an 8-bit checksum of all the hex data on the line. The 8- bit sum of all hex data pairs (including count, address, type, data and checksum) will be zero.

The file is terminated by a hex line with a byte count of zero, address of zero and type of "01".

NOTE: NOTE: The hex file does not need to specify data for every address in the PROM. If there are any gaps in the data then Don’t Know will be assumed for unspecified locations.

PROM Data File Example

Example 1 - 4-input, 4-output PROM

```
:100000000304040204020204040202040204040AB7
```

```
:00000001
```

Example 2 - 8-input, 8-output PROM

```
:200000000F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F00

:20002000060F0E0D0C0B0A0908070F0F0F0F0F0E0D0C0B0A090807060F0F0F0F0F0F0F0F3A

:200040000F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0FC0

:20006000060F0E0D0C0B0A0908070F0F0F0F0F0E0D0C0B0A090807060F0F0F0F0F0F0F0FFA

:200080000F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F80

:2000A000060F0E0D0C0B0A0908070F0F0F0F0F0E0D0C0B0A090807060F0F0F0F0F0F0F0FBA

:2000C0000F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F40

:2000E000060F0E0D0C0B0A0908070F0F0F0F0F0E0D0C0B0A090807060F0F0F0F0F0F0F0F7A

:00000001FF
```

Appendix A—Device Pin Types

Every device pin has a characteristic known as its *pin type*. The pin type is set when the part entry in the library is created and cannot be changed for individual device pins on the schematic.

See the DesignWorks Schematic User's Manual for information on how to set the pin type while creating a device symbol.

What Pin Types are Used For

For many general schematic editing purposes, the pin type will be unimportant and can be ignored. However, pin type settings are important in the following cases:

The pin type of each pin is used by the simulator to select what type of output values are generated by a pin. e.g. An open collector output will not generate a HIGH drive level.

Pin type information is required in many netlist file formats for FPGA layout and digital simulation.

Correct pin type settings allow the ErrorFind tool to check for fanout and multiple drive situations.

Other future analysis tools may use this information for timing and loading analysis.

Pin Types Table

The following table lists the function of each of the pin types available in

DesignWorks. The Output Value Mapping column specifies how output values specified by the model are mapped to actual pin drive values.

Pin Type	**Description**	**Initial Value**	**Output Value Mapping**
Input	Input - this is the default for pins created using the DevEditor tool. This setting is used for all pins on discretes except those with some digital function. No output value can be placed on an input pin.	HIGHZ	No output drive allowed.
Output	Output - always enabled.	DONT01	None
TriState	Output - can be disabled (i.e. high-Z). Note: The three-state capability only exists for specific primitive types that have a three-state enable pin. For other types this will behave like OUT.	DONT01	None
Bidirectional	Bidirectional	DONT01	None
Open Collector	Open collector output - i.e. pulls down but not up	DONT0Z	HIGH maps to HIGHZ.
Bus	Bus pin - This does not represent a physical signal but is a graphical representation of a group of internal pins, each having its own type. Bus pins cannot have values and are not supported on primitive device types.	None	None.
Tied Low	Output - always driving low	LOW	All values converted to LOW.
Tied High	Output - always driving high	HIGH	All values converted to HIGH.
Latched Input	Input to a transparent latch - this is used for calculating cumulative setup and hold times.	HIGHZ	No output drive allowed.
Latched Output	Output from a transparent latch - this is used for calculating cumulative setup and hold times.	DONT01	Same as OUT.
Clocked Input	Input to an edge-triggered latch - this is used for calculating cumulative setup and hold times.	HIGHZ	No output drive allowed.
Clocked Output	Output from an edge-triggered latch - this is used for calculating cumulative setup and hold times.	DONT01	Same as OUT.
Clock Input	Clock input - this is used for calculating cumulative setup and hold times.	HIGHZ	No output drive allowed.
Open Emitter	Open emitter output - i.e. can pull up but not down.	DONT1Z	LOW maps to HIGHZ.
No Connection	A no-connect pin	HIGHZ	No output drive allowed.

Device Pin Type and Simulator Efficiency

Incorrect device pin type settings can have a major impact on simulation speed, even in cases where they wouldn't affect the correctness of the results.

Bidirectional Pins

The use of bidirectional pins should be avoided unless specifically required by circuit logic. On primitive types, any value change on a signal attached to a bidirectional pin will cause the device model to be called to reevaluate the device. On sub-circuit devices, the simulator must make several passes through all circuit levels which may affect the value of the signal or be affected by it. Setting a pin on a sub-circuit device to be an input or output greatly reduces this overhead.

Output Pins

If a device pin will only ever be used to drive the attached signal and the device cannot be affected by changes in value on the pin, then it should be an output type. Changes in the value of a signal attached to an output pin do not cause the device model to be called for reevaluation. This is particularly significant for sub-circuit devices.

Input Pins

Pins with an input type setting can never place a drive value on the attached signal. On sub-circuit devices this provides an important hint to the simulator that internal value changes on the attached signal will not affect any other circuit level.

Appendix B—Timing Text Data Format

When a Copy or Cut operation is done on a selected area in the Timing window, two types of data are placed on the system clipboard:

A picture of the selected area, in standard PICT format.

A text description of the signal value changes occurring in the selected area.

This Appendix describes this text data format.

General Description

The clipboard text format used by the Timing tool is actually a subset of the command structure supported by the Test Vector tool. Thus, it is always possible to paste timing data into the Test Vector. Copying data from the Test Vector tool to the timing diagram is also possible with some restrictions. The Timing tool does not support control commands like $REPEAT/$END which are available in the Test Vector.

Specific Description

The following rules describe the timing text data format:

The data is pure ASCII text with no special binary codes except for standard tab and carriage return characters.

The format of the data is based on the common "spreadsheet" text data format, i.e. each text item is followed by a tab character, except for the last one on a line which is followed by a carriage return.

Every line has the same number of text items on it.

The first line of the text (i.e. up to the first carriage return) is a "header" which indicates the meaning of the items on the following lines, by position.

The lines following the header are signal value lines. Each line represents one time step. A complete data line is written out each time any value on the line changes. No line is written out for time steps in which none of the represented signals changed value.

Header Format

The header consists of a series of "commands", each starting with a $ sign, which describe the meaning of the corresponding data items on the following lines.

The header always contains the command "$T" (denoting a time column), followed by a tab character, followed by "$D" (denoting a delay column). The remaining items depend on the traces that were selected in the timing diagram.

NOTE: The Timing tool always places the time and delay items in the order given here, although it will accept data with them in any order, or even completely missing. Since time and delay are redundant, either one is sufficient. If they are both missing a default delay value will be used.

Single Signal Items

An individual signal is specified by the text $I (for input) followed by a space, followed by the name of the signal. If the signal contains any blanks or control characters, it will be enclosed in quotation marks.

Grouped Items

Grouped items are denoted by the text "$I" followed by a blank, followed by the name of the group, followed immediately (without any spaces) by a list of the signals in the group, contained in square brackets. Any group or signal name which contains blanks or control characters will be enclosed in quotation marks.

NOTE: The Test Vector tool does not use group names, but will ignore any text immediately preceding the "[]" grouping brackets. For this reason, there must not be a space between the group name and the "[" or the group name will be interpreted by the Test Vector as a signal name.

Data Line Format

Each line following the header must contain one data item for each item in the header line. Thus, the first two items will always be:

> The time at which the events on this line take place. The Timing tool places the absolute time at which the events occurred (i.e. corresponding to the time scale on the diagram) in this column. However, when the data is pasted, the times are considered to be relative to the time of the first data line. This is a decimal integer which may take on any 32-bit unsigned value.
>
> The delay from this step to the next step. This is redundant information, since it can be derived from the times in the first column. It is provided for compatibility with Test Vector and for improved flexibility in exporting to outside software systems.

NOTE:

1) If the delay and time columns do not match, the longest time is used.
2) The delay on the <u>last line</u> has special significance because it indicates the delay from the last signal change to the end of the selected interval. When pasting, this value is used to determine how much time to insert.

The following items on a line will be signal or group values matching the items in the header.

> For groups or individual signals, the special values ".X" (for DONT01) and ".Z" (for HIGHZ) indicate that all signals covered by this data item are in the given state.
>
> Individual signals not in Don't Know or High Impedance states will be either "0" or "1".
>
> Grouped signals which are not <u>all unknown</u> or <u>all high impedance</u> will be specified by a hexadecimal value. The least significant bit of the value corresponds to the rightmost signal in the group list. Special character "X" may be substituted for a hex digit if any of the four signals represented by that digit is unknown, or "Z" if all the signals represented by that digit were high impedance.

Timing Text Example

Following is an example of the timing text data and the corresponding timing diagram:

$T	$D	$I Q[Q0 Q1 Q2 Q3]	$I SI	$I LD	$I CLK
87410	2	F	1	0	0
87412	10	F	1	0	1
87422	10	F	0	0	0
87432	1	F	0	0	1
87433	9	E	0	0	1
87442	10	E	0	0	0
87452	1	E	1	0	1
87453	9	D	1	0	1
87462	10	D	1	0	0
87472	1	D	1	0	1
87473	9	B	1	0	1
87482	10	B	1	0	0
87492	1	B	1	0	1
87493	9	7	1	0	1
87502	4	7	1	0	0

Appendix C—Simulation Attribute Fields

The tools making up the Simulator package make use of a number of attribute fields to store device, signal and pin parameters and simulation setup information. The following table summarizes the usage of these fields.

The following information is provided in the attribute table:

Field Name	The name of the field as it appears in the Define Attributes dialog.
Type	The type of objects this field is used in.
Tool	The name of the tool that uses this field. If this is blank, then the field is provided as a convenience for the user but not "hard-wired" to any program function.
Inst/Def	Specifies whether the field data is kept with the definition or instance of a sub-circuit. I.e. Instance fields can have a different value in each copy of a sub-circuit.
Description	How the field is used.

Field Name	**Type**	**Tool**	**Inst/Def**	**Description**
ABELSrcName	Device	FromABEL	Def	Contains the name of the ABEL source file used to generate this part definition.
Delay.Dev	Device	Sim	Def	Specifies device delay. For most devices, a single decimal integer 0 to 32,767. For Clock, One Shot and SetupHold devices, two integers separated by commas. Should be set using the Simulation Parameters command and not edited manually.

Delay.Dev.Max	Device		Def	Maximum device delay. This is provided for use with the Load From Attribute button in the Simulation Parameters command. Not used internally.
Delay.Dev.Min	Device		Def	Minimum device delay. See Delay.Dev.Max.
Delay.Dev.Typ	Device		Def	Typical device delay. See Delay.Dev.Max.
Delay.Pin	Pin	Sim	Inst	A decimal integer specifying pin delay in the range 0 to 32767. Should be set using the Simulation Parameters command and not edited manually.
Delay.Pin.Max	Pin		Def	Maximum pin delay. See Delay.Dev.Max.
Delay.Pin.Min	Pin		Def	Minimum pin delay. See Delay.Dev.Max.
Delay.Pin.Typ	Pin		Def	Typical pin delay. See Delay.Dev.Max.
DWLSrcName	Device	FromABEL	Def	Contains the name of the ".dwl" file used to generate this part definition.
DWLSrcPath	Device	FromABEL	Def	Contains the "path name" of the ".dwl" file used to generate this part definition, i.e. names of all nested directorys.
ExtCctDate	Device	SimLoad/ Schematic	Def	The "last modified" date of the external circuit file, stored as an integer in 1/60 sec since Jan. 1, 1904. Should not be edited manually. See ExtCctName.
ExtCctName	Device	SimLoad/ Schematic	Def	The file name of the external circuit file. Should normally be set using the SimLoad tool or Attach External Sub-circuit command.
ExtCctPath	Device	SimLoad/ Schematic	Def	The "path name" of the external circuit file, i.e. names of all nested directories. See ExtCctName.
Initial.Pin	Pin	Sim	Inst	The initial value for the pin on Clear Simulation. Output pins only. One character: 0, 1, X or Z.
Initial.Sig	Signal	Sim	Inst	The initial value for the signal on Clear Simulation. One character: 0, 1, X or Z.
Invert.Pin	Pin	Sim	Def	Any non-empty value indicates pin inversion.

Sim.InputMap	Design	Sim	Def	Used to specify alternate input mapping for primitive devices. Must be exactly five characters, each one 0, 1 or X. First character specifies the value to substitute for a low input value, second for a high input value, third for high impedance, fourth for Don’t Know, fifth is Conflict. e.g. “011XX” will substitute a high value for high impedance. Default is “01XXX
Sim.TrigSave	Design	Sim	Def	Used by the Simulator to store the current trigger setup. Should not be edited manually.
TestVectors.Cct	Design		Def	Used to store sets of test vectors for the design. Not used internally.
TestVectors.Dev	Device	Test Vector	Def	Used to store test vectors for the device. Loaded and saved by the Place and Test Device and Save Test to Device commands in Test Vector.
Timing.Save	Design	Timing	Def	Used to save the current status of the timing window. Should not be edited manually.

Appendix D—Primitive Device Pin Summary

In DesignWorks primitive device types, the function of each pin is determined by its type (i.e. input or output) and its sequential position in the device's pin list (i.e. as seen when the part is opened in the DevEditor). Pin name is not significant. Each type has specific rules about the ordering of pins. Failure to adhere to these rules will result in incorrect simulator operation.

NOTE: Bus pins are not supported on primitive device types.

For many primitive types, certain control inputs and outputs can be omitted to create simplified device types. E.g. On flip-flop types, the Set and Reset inputs can be omitted. The table below shows which combinations of inputs are allowable and the required order.

Pin Inversion

In addition to the pin function options shown in the table, any pin on any device can be inverted by specifying a value in the Invert.Pin attribute field. Any non-empty value will cause the pin logic to be inverted.

Pin Function Table

The following table lists the pin functions and orders for all primitive device types. In some cases, a number of pins can be optionally omitted, so rules are given rather than enumerating all possible combinations.

See Chapter 11—Primitive Devices on page 129 for more information.

Primitive Type	Limitations	Pin Names and Types	Possible Pin Orders
NOT	Exactly 1 input and 1 output	IN - in OUT - out	1) IN OUT
AND, NAND, OR, NOR, XOR, XNOR	N inputs: 1 <= N <= 799	$IN_0..IN_{N-1}$ - in OUT - out	1) $IN_0..IN_{N-1}$ OUT
X-Gate	Exactly 2 ports and 1 enable	X1 X2 - bidir EN - in	1) X1 EN X2
Three-State Buffer	N data inputs N data outputs 1 <= N <= 256	$IN_0..IN_{N-1}$ - in $OUT_0..OUT_{N-1}$ - out EN - in	1) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ EN 2) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$
Resistor	Exactly 2 pins	X1 X2 - bidir	1) X1 X2
Multiplexer	L select inputs M output bits 1 <= M <= 256 N inputs/ output 2^{L-1} <= N <= 2^L i.e. the number of inputs per output bit can be less than the number of select input combinations	$S_0..S_{L-1}$ - in $IN_{0,0}..IN_{N-1,M-1}$ - in * EN - in § $OUT_0..OUT_{M-1}$ - out * $IN_{n,m}$ is the input routed to output m when select value is n. § An enable input can exist only if N == 2^L, otherwise the extra input is assumed to be a data input.	1) $IN_{0,0}..IN_{0,M-1}$ $IN_{1,0}..IN_{1,M-1}$ .. $IN_{N-1,0}..IN_{N-1,M-1}$ $S_0..S_{L-1}$ $OUT_0..OUT_{M-1}$ 2) $IN_{0,0}..IN_{0,M-1}$ $IN_{1,0}..IN_{1,M-1}$ .. $IN_{0,0}..IN_{N-1,M-1}$ $S_0..S_{L-1}$ EN $OUT_0..OUT_{M-1}$ § § Option 2 only if N == 2^L
Decoder	L select inputs M output bits 1 <= M <= 256 AND 2^{L-1} <= M <= 2^L i.e. the number of output bits can be less than the number of select input combinations	$S_0..S_{L-1}$ - in EN - in $OUT_0..OUT_{M-1}$ - out	1) $OUT_0..OUT_{M-1}$ $S_0..S_{L-1}$ 2) $OUT_0..OUT_{M-1}$ $S_0..S_{L-1}$ EN

Adder, Subtracter	N output bits N "A" operand inputs required N "B" operand inputs optional 1 <= N <= 256	$A_0..A_{N-1}$ - in $B_0..B_{N-1}$ - in CIN - in $SUM_0..SUM_{N-1}$ - out COUT - out	1) $A_0..A_{N-1}$ $B_0..B_{N-1}$ $SUM_0..SUM_{N-1}$ CIN COUT * * $B_0..B_{N-1}$ CIN & COUT can be omitted in any combination
D Flip-Flop, D Latch	Must have at least D and CLK inputs and Q output	S - set in D - D in C - clock in R - reset in Q - out NQ - inverted out	1) S D C R Q NQ 2) S D C R Q 3) D C R Q NQ 4) D C R Q 5) D C Q NQ 6) D C Q
JK flip-flop	Must have at least CLK input and Q output	S - set in J - J in K - K in C - clock in R - reset in Q - out NQ - inverted out	1) S J C K R Q NQ § 2) S T C R Q NQ *§ 3) T C R Q NQ *§ 4) C R Q NQ § 5) C Q NQ § § NQ can always be omitted * T = J & K tied together
Register	N output bits N input bits 1 <= N <= 256	$IN_0..IN_{N-1}$ - in CLK - in CLR - in $OUT_0..OUT_{N-1}$ - out	1) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK CLR 2) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK

Counter	N output bits N input bits (optional) 1 <= N <= 256	$IN_0..IN_{N-1}$ - in CLK - in LD - in CLR - in UP - in EN - in $OUT_0..OUT_{N-1}$ - out COUT - out	1) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK LD CLR UP EN COUT 2) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK LD CLR UP COUT 3) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK LD CLR COUT 4) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK LD COUT 5) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK COUT Note: $IN_0..IN_{N-1}$ & COUT can always be omitted
Shift Register	N output bits N input bits 1 <= N <= 256	$IN_0..IN_{N-1}$ - in CLK - in LD - in CIN - in $OUT_0..OUT_{N-1}$ - out	1) $IN_0..IN_{N-1}$ $OUT_0..OUT_{N-1}$ CLK LD CIN 2) $OUT_0..OUT_{N-1}$ CLK CIN
One Shot		CLK - in CLR - in Q - out NQ - out	1) CLK CLR Q NQ 2) CLK CLR Q
Clock Osc	Exactly one output pin	CLK - bidir	1) CLK
Binary Switch	Exactly one pin	SW - bidir	1) SW
SPST Switch	Exactly 2 pins	X1 X2 - bidir	1) X1 X2
SPDT Switch	Exactly 3 pins	X1 X2 COM - bidir	1) X1 X2 COM
Binary Probe	Exactly 1 pin	PR	1) PR
Hex Keyboard	4 or 5 pins	$X_0..X_3$ - bidir STROBE - out	1) $X_0..X_3$ STROBE 2) $X_0..X_3$
Hex Display	Exactly 4 pins	$X_0..X_3$ - in	1) $X_0..X_3$
SetupHold	Exactly 3 pins	CLK - in DATA - in Q - out	1) CLK DATA Q

Unknown Detector device	Exactly 2 pins	D - in Q - out	1) D Q

Index