

Introduction to LogicWorks (Version 5)

January 24, 2005

by: Kevin Su

0. INTRODUCTION

These notes are meant as a supplement for students taking CS 2513 (Computer Organization I), especially for those who lack experience in the use of Logic Works. Logic Works provides the student with a great deal of flexibility in the design and simulation of the digital circuits that he or she will implement using the chip set and trainer kit. Since the design and simulation steps are meant to verify the correct operation of a circuit before it is implemented, and as a result, save valuable time.

If you have any questions about any topic that is, or is not, covered herein, or you believe that something mentioned may be in error, please contact me with your questions or concerns. My current e-mail address is xsu@cs.utsa.edu

Note: Unless mentioned, all mouse-click are left clicks. "Double-click" and "click-and-hold" should be performed as indicated.

1. TOOLS

1.1 Drawing Window

The **Drawing Window**, also called the **Design Window**, will contain the circuits you design, and is your main interface with the software. Circuits in the Drawing Window act in concert with timing information in the **Timing Window**, as will be seen later.

1.2 Parts Palette

The **Parts Palette**, or the **Library**, contains all of the parts that you will use to design circuits, and some parts that you will not use. What you use will depend on the course you are taking, and your own ingenuity in designing circuits.

1.3 Timing Window

When active, the **Timing Window** will display the waveforms associated with your circuit. This is the other window where important things happen.

1.4 Drawing Toolbar

The **Drawing Toolbar** is used to control the **Cursor** and modify the status of the Drawing Window. The options of the Drawing Toolbar can turn the Cursor into a magnifying glass, a logic probe, a precision deletion tool, a labeling tool, and a wiring tool. It can also be used to cut, copy, paste, and obtain information about selected parts.

1.5 Simulation Toolbar

The **Simulation Toolbar** is used to control the Timing Window. The **Simulator** can be used to control the rate at which simulations are generated, or to step through simulations in incremental time steps. The Simulator options can be accessed through the **Simulation** menu of the Command Toolbar.

2. BASIC DESIGN

2.1 Part Placement

To **place** a part in the Design Window, double-click on the desired part name in the library. A shadowed version of the part will appear at the cursor. Place the part at the desired location and click.

The part selected will remain active for placement until it is deactivated. If you want to select a new part, double-click on that part name and repeat the previous steps. If you just want to deactivate the part without selecting a new one, press the space bar or the Escape key.

Users of Logic Works 3 may recall that the Palette contained an **Orientation Tool**. This function has been moved to the **Schematic** menu of the Command Toolbar. When the Orientation command is selected, the user may select from among the eight part orientations offered on the graphic menu. Once an orientation is selected, all parts inserted into the diagram will appear having that orientation.

2.2 Wiring

To make **connections** between logic devices, click and hold at the end of an input or output pin of one of the devices, or at an existing node of the wire network. A node is created in a network at any point where a wire branches to form multiple connections. Move the cursor to the point where you want the wire to end. This can be another pin, another wire, or an empty location. Be careful, because this method of wiring can produce undesired nodes at points where wires cross.

Another method of wiring involves the use of the wiring drawing tool on the Drawing Toolbar.

2.3 Editing

To **move** a part or a wire, click on it and drag it to the desired location. Selected parts will appear highlighted in black, and wires will appear highlighted in yellow. Be careful when moving parts, because wires attached to moving parts will “stretch,” and unwanted nodes may be created.

To move a large number of parts, click and hold the mouse button at an empty location and drag the mouse until all of the parts you wish to move are contained (at least in part) in the window that has been created. When you release the mouse, all parts that can be moved will appear highlighted. Click and hold the mouse on any of the highlighted parts, and they will all move together. Again, be careful, because wires may cross when you move parts.

To **delete** a part or wire, click on it and press the delete key. To delete a large number of parts, select them all as before (click and hold, then drag) and press the delete key.

If you get in trouble, the **Undo** command in the **Edit** menu has a very good memory.

3. THE DRAWING TOOLBAR

As mentioned earlier, The **Drawing Toolbar** is used primarily to change the function of the cursor. It allows you to change attributes such as part placement and wire type, make corrections in circuits, insert text, change the view of the design window, and perform logic probes of your circuit.

The Drawing Toolbar is most easily thought of as having five parts: **File Management, Part Movement and Modification, Cursor Modification, Drawing Window Modification, and Part Type**.

3.1 File Management

The first four buttons of the Drawing Toolbar are used for **file management**. Starting from the left, these four buttons are used to **Create** new files, **Open** an existing document, **Save** the active document, and **Print** the active document. Each of these commands, as well as the remainder of the file management commands, can be found in the **File** menu of the Command Toolbar.

3.2 Part Modification

The next five buttons of the Drawing Toolbar are used for **part movement and modification**. Continuing from the **Print** button, these buttons are used to **Cut** parts which have been selected on the Drawing Window, **Copy** selected parts to the Clipboard, **Insert** the contents of the Clipboard into the Drawing Window, **Duplicate** the selected parts and make them available for placement using the cursor, and **Get** and **Set Information** about the selected part. In the case of the last command, only one part can be selected at a time.

3.3 Cursor Modification

The next seven buttons of the Drawing Toolbar are used for **Cursor Modification**. These functions are probably the most useful of those on the Drawing Toolbar. You will find some more useful than others, but each one will be covered in greater detail here.

3.3.1 Magnify

The **Magnification** cursor will present a centered and magnified view of the location that is clicked. Magnification can also be done using the **Magnify** command under the **View** menu.

There is no **Reduce** cursor, but reduction can be done using the **Reduce to Fit** command under the **View** menu, or through various zooming techniques, all of which will be discussed later.

3.3.2 Probe

The **Probe** tool has two uses. First, when the cursor is touched to a pin and held (click and hold), that pin's logic level is displayed.

Second, if you enter a logic value while probing a pin, the pin will take on that value. Be careful when setting values, as logic conflicts can occur which are not immediately detectable. You should not use this function unless you are an experienced user of Logic Works, and even then, you will probably not find it that useful.

3.3.3 Pointer

This is the default cursor selection. When enabled, the cursor will function in all of the ways previously described.

3.3.4 Zap

In several cases, you will want to delete one part of a wire without deleting the whole mesh, or you may want to delete one object in a particularly tight space. The **Zap** cursor allows you more flexibility in deleting objects. When enabled, clicking on an object with the zap cursor will delete only the part of an object that appears at the tip of the cursor. This is very useful for removing errant parts of wires.

3.3.5 Text

Using the **Text** cursor, you can insert text into schematics, i.e. titles, chip descriptions or wire and pin labels. Clicking on the text button changes the cursor into a pencil. When the pencil tip is inserted in a random location, a text window appears at that location, and miscellaneous text can be inserted. Type in the desired text, and click the text cursor elsewhere once you are finished. To edit text, click the text cursor in the vicinity of existing text.

When the pencil tip points at a pin, wire, or part, you can name that object. When labeling a pin, click the pencil tip on the pin, and a window will open. Type the desired text in the window, and click the mouse or hit Return. The text will appear in blue above the pin.

When labeling a gate, click the pencil tip on the gate, type the desired name into the window which opens, and click the mouse or hit Return. The name will appear in red above the gate.

When naming a signal, click the pencil on the end of a pin or on the wire which will carry the signal. Type the signal name, and click the mouse or hit Return. The name will appear in red at some point along the signal. If the **Add Automatically** option is selected in the **Simulation** menu, the signal name will also appear in the **Timing Window**.

3.3.6 Wiring

Two palette tools are available for wiring: normal wire, which can be placed using the **Normal cross**, and bus wire, which can be placed using the **Bold cross**. Bus wire is not used frequently in digital applications, but be aware that any wire connected directly to a bus becomes a bus, even if the normal wire cursor is selected.

Unlike wiring done with the normal cursor, wiring done with the wiring cursor does not have to begin at a pin.

3.4 Drawing Window Modification

The next four buttons of the Drawing Toolbar affect what is displayed on the Drawing Window. Continuing from the **Bus Wire** button, the next four buttons cause the design page to **Zoom Out**, cause the design page to **Zoom In**, **Fit** the design page to the Drawing Window, and display the parts in the design page at **Normal Size**.

3.5 Part Type

The last two buttons control the body of parts that are used in circuit design. The first button opens the **PROM / RAM / PLA Wizard**. Of the three features of this option, the only option which might be useful to 3504 students is the Programmable Logic Array (PLA) construction tool. For more details on the use of this option, consult Chapter 10 of the Logic Works 4 manual.

The second button opens and closes the **Parts Palette**.

4. LIBRARIES

In most circuit design, there are only a few libraries with which you will need to concern yourselves.

4.1 Simulation Gates.CLF

This library contains individual logic gates: **AND**, **OR**, **NOT**, **NAND**, **NOR**, **XOR**, and **XNOR**. Depending on the type of gate, different numbers of inputs are available. You will notice that several of the gates have inverted inputs available, but from the standpoint of determining chip costs and gate delays, it is not a good design practice to use those gates, assuming that you use primitive gates to design circuits in the first place.

4.2 Simulation IO.CLF

These parts represent individual input/output devices. Of the devices which are only found in this library, three sets of devices may be of use to designers.

4.2.1 Seven-Segment Displays

There are multiple seven-segment displays in the I/O Library. Of those, the ones called **7-Seg Disp Inv - (Color)** operate in the same way as the seven-segment display included in the 2513 kit: when a logic **zero** is applied to one of its pins, the corresponding segment becomes lit, and when a logic **one** is applied, the corresponding segment stays unlit. Be sure to keep this in mind when you perform a design using a seven-segment display. The other seven-segment displays in this library have segments which light on logic one; these are **not** the devices you should use.

4.2.2 LEDs

The LEDs in this library can be thought of and used in the same way as the LEDs in your trainer kit. To correctly wire the LEDs, ground the pin which is connected to the bubble, and connect the other pin to the output device which you are using to drive the LED. When properly grounded, LEDs will light on logic one.

4.2.3 Pushbuttons and Switches

In addition to the Binary Switch (which will be discussed later), the I/O Library contains a Single-Pole Double-Throw (SPDT) pushbutton, a SPDT switch, and a Single-Pole Single-Throw (SPST) switch. An SPST switch is one which is either open or closed. A SPDT switch can take on two different logic levels, depending on what connections are made to the switch inputs. A SPDT pushbutton is like a SPDT switch, except it normally takes on one value, and only goes to its other value when momentary contact is made. For double throw switches, make sure that both inputs are connected to circuit elements which can take on known values.

4.3 Simulation Logic.CLF

The parts in this library are individual higher-level logic devices, such as adders and registers. Students in 2504 and 3504 may find these devices useful:

4.3.1 Adder

The Logic Library contains a number of adders. To simulate the adder included in the 2513 chip set, you should select the part called **Adder-4**. This adder has the same input, output, and carry pins as your Adder chip.

4.3.2 Flip-Flops

The individual **D flip-flops** operate in the same manner as the D flip-flops in your chip set. The **D** input is the same in Logic Works as it is on your chip. The **C** input represents the **clock** pin. The **Q** and **Q'** outputs represent the standard and complemented outputs. The **S** and **R** lines represent the **set** and **reset** (clear) lines of the flip-flop. Note that these pins are active-low, as represented by the bubbles on their inputs. If you know that you will never have to preset the flip-flop, connect the S input high. To implement a clear function into a sequential circuit, connect the R input to a binary switch (See 4.5.2: Binary Switches.) whose value is high (logic 1) for the time in which you do **not** want the circuit to be reset.

The pins on the **JK flip-flop** follow the same conventions as the D flip-flop.

4.3.3 Multiplexers

This library contains a number of multiplexers. To simulate the multiplexer included in the 2513 chip set, your best bet is to use two of the **Mux-4** devices. The names of the pins on this device are much the same as those on your multiplexer chip.

4.4 7400DEVS.CLF

These parts represent devices in the **7400 series of integrated circuits**, and correspond to the chips that you purchased. For reasons of convenience, most of the design that you do in CS 2513 will be simulated in Logic Works using 7400 devices instead of chips.

- The 7400 devices have pin numbers that correspond to the actual integrated circuits. While the pin locations do not match exactly, a well-laid out design that uses 7400 devices can save time when the time comes to wire a circuit.
- Having the 7400 devices laid out saves time when preparing Whole Chip Costs and Fractional Chip Counts for reports.

4.5 DEMOLIB.CLF

This library contains miscellaneous parts, of which five in particular are very useful.

4.5.1 +5V, Ground

Just as you should never keep a input pin floating when wiring a chip, it is a good design practice to wire input pins high or low if you know that they will always have a certain logic value. These two signals can be used to provide constant logic levels of 0 or 1 to the various pins of your circuit. In Logic Works, 7400 devices do not have power or ground pins, but there are times when you will want to keep reset pins, enable pins, clear pins, or input pins at logic 1 or logic 0.

4.5.2 Binary Switches

The 2504 and 3504 kits contain banks of DIP switches, which can place a logic 1 or logic 0 on a given pin. In Logic Works, **Binary Switches** work the same way. A Binary switch is a special kind of SPDT switch whose inputs are already connected to logic 1 and logic 0. The designer has only to wire the output of the switch to an input pin or named signal, and toggle the value by clicking on the switch. If the switch does not toggle, make sure that the simulation is active by pressing the **Reset** and **Run** buttons on the Simulation Toolbar.

Binary switches (and SPDT pushbuttons) make good manual clocks; they work the same way as the push-button switches in your lab kits.

4.5.3 Binary Probes

When debugging a physical circuit, it is a good practice to use a wire and an LED to observe the logic levels within the circuit. Connect one end of the wire to the LED, and use the other end to probe the output. **Binary probes** work the same way in Logic Works. They are better than the **Probe** on the **Palette** in that once they are placed, they remain in the circuit, and are updated whenever the circuit changes.

Multiple probes can be placed at different locations in a circuit, allowing you to troubleshoot many locations at once.

The values displayed on the probes follow the rules of Boolean algebra, but conditions may exist such that the probe displays a non-numeric condition.

- **(X)** Don't know/Don't care condition: An output is at an unknown state.
- **(Z)** High impedance/No input condition: The probe is not connected to an input.
- **(C)** Conflict condition: An output is being forced to conflicting logic levels.

In general, these conditions are not desirable, and you should check your circuit for errors should one occur.

4.5.4 Clocks

Although it may seem strange, the clock in this library will not be very useful when you need a clock in your circuit. In the case where you are changing the inputs of a circuit using binary switches, a clock oscillates much too rapidly to be of any use. In cases like these, you will want to model the clock in much the same way that it exists in your trainer kit: as a push-button switch. (i.e., another binary switch or a SPDT pushbutton) In the case where you are using a **Timing File** to drive the inputs, it can be very difficult to synchronize the start of the clock with the start of the timing file. In these cases, the best thing to do is to drive the clock from the timing file as if it were another input. (See 5.2.2: **Automatic Simulation** for more information on the details of Timing Files.)

The best use for clocks that I have found is in generation of timing files for use in combinational logic circuits. Suppose that you have a circuit whose inputs are one of n Boolean variables. If you wish to compare the output of your circuit with the information contained on a truth table, you would need to write a timing file that contains 2^n lines of information. If your circuit contains more than 4 input variables, the creation of such a timing file can become tedious and prone to error.

To generate such a timing file, you can build an n -bit binary counter, which requires n clocks. Place these clocks in the Design window, and give each one a name corresponding to the variables of your circuit. This is important, as timing files only work when the names of the signals contained in the file correspond exactly to the names of the signals in a circuit.

To make the counter work, each variable (or clock) must have a frequency that is half that of the variable which immediately precedes it. For example, the clocks have default high and low times of 10 nanoseconds. You can let that be the high and low time of the clock that represents your least significant bit. Now, each clock after the first must have high and low times which are **double** that of the clock before it. Remember: to halve the frequency, double the period. To change the timing parameters of the second clock, select it by clicking on it. Then choose the **Simulation Parameters** option under the **Simulate** menu bar. A window will appear allowing you to modify the clock parameters. Give the second clock high and low times of 20 nanoseconds. Then, as stated earlier, repeat this for each clock, progressively doubling the high and low times for each clock.

Once you have done this for all of the clocks, you can use the **Single Step** button on the Simulation Toolbar to step through a full count cycle (all zeros to all ones). If you named your clocks correctly, this information will appear in the timing window. Follow the directions given below for **exporting** timing files (See 5.3 **Importing and Exporting Timing Files.**) Now, when you need a timing file that tests all of the inputs of a combinational circuit, assign names to the inputs of the circuit that match the signal names in the timing file. Then, follow the directions below for **importing** a timing file. Use the timing file you created to drive the circuit, and you will end up with a timing file showing the correct inputs and outputs.

5. SIMULATION AND TIMING

5.1 The Simulation Toolbar

The **Simulation Toolbar** is your interface with the Simulator and the Timing Window. Its features give you control of the speed and status of the simulation, the status of the various signals of the circuit, and the size of the Timing Window. The Simulation Toolbar is best thought of as having three parts: **Signal Status**, **Window Status**, and **Simulator Status**

5.1.1 Signal Status

The first seven buttons control the status of signals, and to a degree, their appearance in the Timing Window. Starting on the left, these buttons function as follows:

5.1.1.1 Toggle

The first button **Toggles** the presence and absence of the Timing Window.

5.1.1.2 Add Signals

The second button **Adds Selected Signals** to the Timing window if the Add Automatically option of the Simulation menu is not already selected.

5.1.1.3 Triggers

This button opens the trigger option menu.

5.1.1.4 Simulation Parameters

This button can be used to carry out the same functions as those which can be performed using the **Simulation Parameters** command under the Simulation menu.

5.1.1.5 Stick Signals

This button can be used to **Stick Signals** at a given logic level. A stuck signal is one which remains at one logic level despite changes which might be affecting.

5.1.1.6 Reset Simulator

This button **Resets the Simulator**. The Timing Window is cleared of all waveforms, and regardless of the current time indicated in the Simulation Time Window, the next time step taken will be time zero.

5.1.1.7 Clear Unknowns

This button **tries to clear** all signals which have unknown values. Sometimes this is not possible, as a part may have undriven inputs, which by definition provide **Z** inputs to the part.

5.1.2 Timing Window Control

The next three buttons control the scale of the Timing Window. These buttons **Zoom In** on the timing scale (the Timing Window contains less time), **Zoom Out** on the timing scale (the Timing Window contains more time), and set the timing scale to its **Normal (Default) Size**.

5.1.3 Simulator Speed

The **Simulator Speed** control consists of three buttons, a sidebar, and a counter. Starting on the left, these controls have the following functions:

5.1.3.1 Single Step

As mentioned previously, the **Single Step** button can be used to a single step in the simulation of a circuit. A step is defined as the shortest amount of time which is required to observe a change in the simulation. As a result, a step can be of any size, and usually reflects the time length of the quickest-changing variable, or of the shortest delay path.

5.1.3.2 Stop Simulator

When the simulator is running, the **Stop Simulator** button stops the progress of the simulation. It does NOT reset the simulation; while stopped, the simulation can be resumed by pressing the Run button or by selecting a Simulation Speed on the sidebar. While stopped, the simulation can also be advanced in single steps.

5.1.3.3 Simulation Speed

The sliding toolbar can be used to **Change the Speed** of simulation. This is most visibly reflected in the speed of the waveforms in the Timing Window. The simulation speed can be changed during simulation, or it can be set while the Simulator is stopped. When the indicator is slid all the way to the left, the Simulator is stopped.

5.1.3.4 Run Simulator

Pressing the **Run** button causes the Simulator to run at its highest speed. This is true whether or not the Simulator is already running.

5.1.3.5 Simulation Counter

This window shows the **Current Time Step**, i.e., the amount of simulation time which has passed since the last reset. The time shown in this window reflects the time shown in the timing window, and the simulator speed selected on the sliding toolbar.

5.2 The Timing Window

When active, the **Timing Window** will display all changes in all named signals. Events that cause signals to change include toggling binary switches, clock pulses, and timing file inputs. These types of changes can be characterized as static (manual) or dynamic (automatic).

5.2.1 Manual Simulation

Switches and probes can be used in the manner described above to achieve a degree of manual simulation. Since Logic Works is an event-driven simulator, a change in any input signal causes all named signals to be updated. As a result, a change in any of the signal values, say, through the toggling of a switch, will cause all of the observable values to change. These changes manifest themselves primarily in two ways. Either all of the binary probes in our circuit will be updated, or the waveforms in your timing window will be updated.

5.2.2 Automatic Simulation

Having to toggle a large number of binary switches in a complex circuit can become tedious, lead to error, or just end up taking a lot of time, especially if you lose your place and have to start over. Fortunately, Logic Works is equipped with the means to automate the simulation process, through the use of **timing files**. A timing file is a compact method of telling the Simulator what signal you want to apply to a given input at a given point in time.

A typical timing file requires at least three headers and at least one row of data, with one element of each row corresponding to each header. Those headers are:

- **\$T**: This header gives the current point in time. Time in Logic Works is generally measured from $T = 0$, and each time step can be thought of as one nanosecond
- **\$D**: This header gives the delay or difference between the time step of the current row and that of the next row. While this information is redundant given the information provided by the first header (and can therefore be left out), you will see this in any timing file which is generated by Logic Works.
- **\$I**: This header denotes each signal that is being driven by the timing file, and should be immediately followed by the name of the signal. When timing files are **exported**, all signals, input and output, will be denoted by the header **\$I**, followed by the signal name

Keeping in mind what each header means, we can see that each row of data corresponds to a point of time (**\$T**), where each signal will have a given value (**\$I**) for a given amount of time (**\$D**).

5.3 Importing and Exporting Timing Files

Once you have created a timing file in an application such as Notepad, you will want to use it drive the inputs of your circuit. To do this, you must **import** the timing file into the Simulator. Select the **Simulation** menu of the Command Toolbar, and select the option called **Import Timing**. Select the name of your timing file, and a broken-line version of the inputs will appear in the timing window.

When you press the Run button on the Simulation Toolbar, Logic Works will cycle through these inputs, applying them at the correct locations. Any outputs that are named in the circuit and appear in the Timing window will also be updated, and their waveforms generated.

Again, in order for a timing file to be successfully imported, all of the signals named in the timing file must correspond to signals that are named in the circuit diagram. Also, the data contained in each row must all correspond to a given header. It is a good idea to place one tab stop between each header, and one tab stop between each piece of data in a row. Since most tab stops default to eight characters, and the header **\$I** takes up three of those characters (including the space), it is a good idea to keep the names of your signals to three characters or less. A signal name should be long enough to distinguish it from others, but not so long that the tabs and spaces turn your timing file into a jumble of letters and numbers.

Once you have obtained a set of correct outputs, you may be asked to provide these outputs in a tabular or textual form, you can do this by **exporting** the timing file. Select the **Simulation** menu of the Command Toolbar, and select the option called **Export Timing**. Give the file a name, and the information contained in the Timing window will be saved in a format similar to timing files that you write yourself.

5.4 Delays

Unless modeled otherwise, devices in Logic Works are treated as nearly ideal devices, having a propagation delay of one nanosecond between the input and the output. Such is not the case in real life; according to most datasheets, a typical device such as a 7400 chip, or a dual-input NAND gate, has a worst-case delay of 22 nanoseconds between a change in the input and the corresponding output. There are times when you will be asked to model this delay in your simulations, in order to meet certain timing or frequency criteria. Fortunately, Logic Works gives you a way to do this.

To **change the delay in one or more gates**, hold down the Control key, and click on all of the devices for which you want to change the delay. Then, under the **Simulation** menu, select the **Simulation Parameters** option. A window will appear, showing the current shortest and longest delays for the selected gate or gates. For all parts, this defaults to one nanosecond. To change the delay, simply type in the delay that you want the selected gate or gates to have. The delay that you type in will be applied to all gates which you select in the Design window, so be sure that all of the gates which you select have the same time delay.

To **change the delay in a chip** (i.e., a 7400 device), you must first unlock the subcircuit containing the gates. Right-click on the device and select **Device Info**. A window containing various information on the selected device will appear. The dialog box labeled **Lock Opening Subcircuit** should be checked. Click on the box to remove the check. Now, you can edit the parameters of the gates inside the chip as if they were individual gates. To do this, double click on the chip you wish to edit. A Design Window containing a gate and pin layout of the IC will appear. As before, select the gates whose timing parameters you wish to edit, and follow the instructions for changing the gate delay.

To save time when changing the chip delays in a large circuit, lay out the entire circuit by placing all of the chips that your circuit requires into the Design Window. You need not wire the circuit in order to implement the propagation delays. For each type of chip in the circuit, select one and perform the procedure for changing the delays of the gates on that chip. When you do this, all of the chips of that type which are already in the design window will have the same delay. Note that this procedure will not work for gates taken from the **Simulation Gates.CLF** library. Instead, you will have to follow the procedure given for changing the delay in multiple primitive logic gates.